

The Virtual Biomembrane: Decoding the Functional Characteristics of Membrane Transporters

Dissertation
zur Erlangung des Grades
des Doktors der Naturwissenschaften
der Fakultät für Mathematik und Informatik
der Universität des Saarlandes

von Andreas Denger

Saarbrücken
2025

Tag des Kolloquiums	09.06.2026
Dekan	Prof. Dr. Jan Reineke
Vorsitz	Prof. Dr. Andrea Volkamer
Erstgutachter	Prof. Dr. Volkhard Helms
Zweitgutachterin	Prof. Dr. Olga Kalinina
Beisitz	Dr. Alexander Gress



UNIVERSITÄT
DES
SAARLANDES

Eidesstattliche Versicherung

Hiermit versichere ich an Eides statt, dass ich die vorliegende Arbeit selbstständig und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe. Die aus anderen Quellen oder indirekt übernommenen Daten und Konzepte sind unter Angabe der Quelle gekennzeichnet. Die Arbeit wurde bisher weder im In- noch im Ausland in gleicher oder ähnlicher Form in einem Verfahren zur Erlangung eines akademischen Grades vorgelegt.

Declaration of original authorship

I hereby declare that this dissertation is my own original work except where otherwise indicated. All data or concepts drawn directly or indirectly from other sources have been correctly acknowledged. This dissertation has not been submitted in its present or similar form to any other academic institution either in Germany or abroad for the award of any other degree.

Saarbrücken, Oktober 2025

gez. / signed

Andreas Denger

Abstract

The identification of membrane transporters and their substrates is crucial for advancing research in metabolomics, pharmacology, and biotechnology. This thesis comprises three published articles that enhance our ability to predict substrates of membrane transporters with machine learning approaches.

First, we developed optimal datasets and feature encodings for predicting transporter substrates across *E. coli*, *A. thaliana*, *S. cerevisiae*, and human. We show that single-organism classifiers can yield robust performance, with a four-class model for *A. thaliana* achieving an F1-score of 0.97. On a binary meta-dataset of all four organisms, the mean F1 was 0.91.

Second, we created an optimization algorithm that employs machine learning to reduce redundancy in functional annotations of transporters, resulting in more manageable datasets. For a meta-organism dataset, this pipeline narrowed the number of classes from 695 to 49, while still covering 96% of transport proteins and achieving a median F1 of 0.92.

Third, we leveraged novel deep learning techniques, including protein language models and 3D structural models, to enhance substrate prediction accuracy. We demonstrated improved classification performance for carriers in *A. thaliana* and for human ion channels.

These studies, and their accompanying software packages, provide a comprehensive framework for identifying and classifying membrane transporters, paving the way for future research and applications.

Zusammenfassung

Die Identifizierung von Membrantransportern und ihren Substratmolekülen ist entscheidend für die Metabolomik, Pharmakologie und Biotechnologie. Diese Dissertation umfasst drei veröffentlichte Artikel, in denen Methoden zur Substrat-Vorhersage entwickelt wurden.

Zunächst entwickelten wir optimierte Datensätze und Sequenz-Kodierungen, um in vier Organismen mit maschinellem Lernen Proteine zu klassifizieren. Wir zeigten, dass Organismus-spezifische Modelle akkurate Vorhersagen treffen können. Ein Vier-Klassen-Modell für *A. thaliana* erreichte einen F1-Score von 0,97. Bei einem Meta-Datensatz bestehend aus vier Organismen lag der F1 bei 0,91.

Anschließend entwickelten wir einen Optimierungsalgorithmus, der Redundanzen zwischen den funktionellen Annotationen von Transportern reduziert. Für einen Meta-Organismus-Datensatz verringerte diese Methode die Anzahl der Klassen von 695 auf 49, von denen noch 96% der Transportproteine abgedeckt waren; der Median der paarweisen F1-Scores lag bei 0,92.

Schlussendlich nutzten wir Deep-Learning-Methoden, u.a. Protein-Sprachmodelle und vorhergesagte 3D-Strukturen, um die Genauigkeit der Substratvorhersage weiter zu verbessern. Wir demonstrierten eine verbesserte Klassifikation von Carrier-Proteinen in *A. thaliana* und von menschlichen Ionenkanälen.

Diese Studien, sowie die dazugehörige Software, bieten ein umfassendes Framework zur Klassifizierung von Membrantransportern, und ebnen damit den Weg für zukünftige Grundlagenforschung.

Acknowledgments

First of all, I would like to thank Prof. Dr. Volkhard Helms for supervising my thesis, for giving me room to implement my ideas, and for always providing help and guidance where it was needed.

I also want to thank the current and former members of the Chair of Computational Biology, who gave me constructive feedback and new ideas for my project, in particular: PD Dr. Michael Hutter, Hoang Thu Trang Do, Hanah Robertson, Sudharshini Thangamurugan, Aram Papazian, Markus Hollander, Dr. Nicolas Künzel and Dr. Thorsten Will.

Furthermore, I want to extend my thanks to Prof. Dr. Ivan Bogeski, as well as Prof. Dr. Bin Qu, for allowing me to take part in their research projects, where I gained new skills, and had the chance to acquire further experience and practical knowledge.

Finally, I want to thank my friends and family for their support during my studies.

Contents

1	Introduction	1
1.1	Motivation	1
1.2	Objectives and Outline	4
1.2.1	First author publications included in this thesis	6
1.2.2	Coauthor publications during doctoral studies	7
2	Background	11
2.1	Biology	11
2.1.1	Proteins	11
2.1.2	Membranes	14
2.1.3	Transmembrane transport proteins	18
2.2	Computational tools and data sources	45
2.2.1	Webservices and databases	45
2.2.2	Statistical methods	56
2.2.3	Machine learning methods	61
3	Optimized Data Set and Feature Construction for Substrate Prediction of Membrane Transporters	75
3.1	Abstract	75
3.2	Introduction	76
3.3	Methods	77
3.3.1	Dataset creation	78
3.3.2	Machine learning features	79
3.3.3	Machine learning pipeline	80
3.4	Results	81
3.4.1	Distinguishing amino acid transporters from sugar transporters in <i>A.thaliana</i>	81
3.4.2	Application to other organisms	87
3.4.3	Datasets involving more than one organism	91
3.4.4	Additional substrates	96
3.4.5	Automatic feature combination	96
3.4.6	Comparisons between classification algorithms	98
3.5	Discussion	99
3.6	Data and Software Availability	102
4	Identifying optimal substrate classes of membrane transporters	103
4.1	Abstract	103

4.2	Introduction	104
4.3	Materials and methods	106
4.3.1	Data retrieval and preprocessing	106
4.3.2	Dataset creation pipeline	106
4.3.3	Pairwise similarity scores	107
4.3.4	Greedy algorithm for clustering of GO terms	109
4.3.5	Pipeline overview and complexity	110
4.4	Results	111
4.4.1	Dataset creation pipeline	111
4.4.2	Application of the redundancy reduction pipeline to the meta-organism dataset	111
4.4.3	Comparison of meta-organism results with substrate classes used in previous studies	115
4.4.4	Yeast transporter dataset	116
4.5	Discussion	122
5	Application of Protein Structure Encodings and Sequence Embeddings for Transporter Substrate Prediction	125
5.1	Abstract	125
5.2	Introduction	126
5.2.1	Motivation and Background	126
5.2.2	Membrane Transporter Substrate Prediction with Machine Learning	126
5.2.3	Applications of Deep Learning Models in Bioinformatics	127
5.2.4	Predictive Methods Based on Protein Structures	128
5.2.5	Outline	129
5.3	Methods	129
5.3.1	Data Availability	129
5.3.2	Protein Dataset Creation	129
5.3.3	Gene Ontology Graph and GO Annotations	130
5.3.4	Blast Databases and PSSM Calculation	130
5.3.5	Predicted 3D Structures and 3Di Sequences	130
5.3.6	Machine Learning Features Derived from Amino Acid Sequence	131
5.3.7	Machine Learning Features Derived from 3Di Sequence	132
5.3.8	Embeddings of Protein Sequence and Structure	132
5.3.9	Dummy Feature	132
5.3.10	Support Vector Machine (SVM) Models	132
5.3.11	Deep Neural Network (DNN) Model	133
5.3.12	Automatic Outlier Detection	134
5.3.13	Evaluation Methods	134
5.4	Results	135
5.4.1	Classifying Substrates of Solute Carriers in <i>Arabidopsis thaliana</i>	135
5.4.2	Categorization of Human Ion Channels	140
5.5	Discussion	145
6	Conclusion and Outlook	149
6.1	Summary	149
6.1.1	Introduction and Background	149

6.1.2	Outline	151
6.1.3	Chapter 3: Developing datasets, features and models . . .	151
6.1.4	Chapter 4: Improving data sources and finding optimal substrate classes for any dataset	156
6.1.5	Chapter 5: Can recent advances in deep learning improve our features and models?	160
6.2	Conclusion and Outlook	162
	Bibliography	167
	Appendix	203
	A Supplementary Material for Chapter 2	203
	B Supplementary Material for Chapter 3	209
	B.1 Code and raw data	209
	B.2 Uniprot Keywords	209
	C Supplementary Material for Chapter 4	211
	C.1 Code and raw data	211
	C.2 Steps in the algorithm pipeline	211
	C.3 Parameters of the transporter dataset pipeline	212
	D Supplementary Material for Chapter 5	221

List of Tables

3.1	Effect of sequence clustering on sample count	89
3.2	Method evaluation for all features and organisms	90
3.3	Additional evaluation for organisms with low sample count . . .	90
3.4	Transfer learning with model trained on <i>A. thaliana</i> dataset . . .	91
3.5	Transfer learning with models trained on other organisms	92
3.6	Evaluation of models trained on combined datasets	95
3.7	Detailed model evaluation on the combined dataset	95
3.8	Evaluation of <i>A. thaliana</i> multi-class models	97
4.1	Spearman correlation between F1 scores and similarity measures	118
4.2	Selected GO subset for yeast dataset	119
4.3	GO term subset for yeast dataset when prioritizing abstract terms	120
5.1	Results for all features with DNN model, when distinguishing solute carriers in <i>A. thaliana</i>	141
5.2	Evaluation scores for human ion channels with SVM and DNN models, for all features	144
C.1	Meta dataset protein counts	214
C.2	Meta dataset GO term counts	215
C.3	Yeast dataset pairwise evaluation scores	216
D.1	Results for all features of <i>A. thaliana</i> SVM model	222
D.2	Results for all features of <i>A. thaliana</i> SVM model, after outlier removal	224
D.3	Feature overview	225

List of Figures

2.1	The peptide bond	12
2.2	Axes of rotation in amino acids	12
2.3	Protein secondary structure	13
2.4	The lipid bilayer membrane	15
2.5	Membrane proteins	16
2.6	Crystal structure of ABC complex	20
2.7	Human P-glycoprotein	21
2.8	Structure of a tripartite efflux pump	22
2.9	Mechanism of rotary ATPases	23
2.10	Structure of the sodium-potassium pump	25
2.11	Binding pocket of human GLUT3	27
2.12	Comparison between GLUT binding sites, GLUT mechanism	28
2.13	LeuT transporter and general structure/mechanism	30
2.14	Three different β -barrel pores	32
2.15	Crystal structure of maltoporin	34
2.16	The potassium channel KscA	35
2.17	Structure and mechanism of aquaporins	37
2.18	Structure and mechanism of an electron transporter	39
2.19	The effects of SVM parameters on the decision function	62
2.20	Radial basis feature transformation	66
2.21	Examples of ROC and PR curves	72
3.1	Chapter outline	76
3.2	Membrane transporter dataset pipeline	78
3.3	Outliers in <i>A. thaliana</i> dataset	82
3.4	Sample count and classification performance in <i>A. thaliana</i>	83
3.5	Sequence clustering and sample count in <i>A. thaliana</i> dataset	84
3.6	Feature importance scores in <i>A. thaliana</i> dataset	85
3.7	Hierarchical clustering of combined dataset	93
3.8	Sequence clustering and sample count in combined dataset	94
3.9	Sequence clustering and classification performance in combined dataset	94
3.10	Evaluation of <i>A. thaliana</i> multi-class models	98
4.1	Flowchart of the entire pipeline	110
4.2	Distribution of annotated protein count for membrane transporter GO terms	112

4.3	Median F1 scores for meta-organism dataset at different coverage thresholds, when removing most abstract terms	113
4.4	Subset sizes for meta-organism dataset at different coverage thresholds, when removing most abstract terms	113
4.5	ML models excluded due to low sample count for the meta-organism dataset, when removing the most abstract terms	114
4.6	Overlap heatmap for all transporters in meta-organism dataset .	114
4.7	Number of GO terms associated with more than a specified number of proteins in yeast	116
5.1	Distribution of F1 scores for sugar/amino acid SVM model in <i>A. thaliana</i>	137
5.2	Distributions of evaluation scores for solute carrier DNN model in <i>A. thaliana</i>	140
5.3	Classification of human ion channels with SVM and DNN models	143
A.1	Amino acid residues	204
A.2	Sequence similarity between seven glucose transporters	205
A.3	MSA of GLUT homologues	206
A.4	Pairwise sequence alignment between two GLUT3 proteins	207
A.5	The effects of regularization parameters on a SVM with RBF kernel	208
C.1	Median F1 scores for meta-organism dataset when not removing abstract terms	217
C.2	Subset sizes for meta-organism dataset when not removing abstract terms.	217
C.3	GO terms that were excluded in meta-organism dataset due to low sample count, when not removing abstract terms.	218
C.4	Overlap percentage heatmap for the meta-organism dataset, when not removing the most abstract GO terms.	218
C.5	Overlap percentage heatmap for the meta-organism dataset, after removing the most abstract GO terms.	219
C.6	GO terms excluded in yeast dataset due to low sample count, at different thresholds	219
C.7	Median F1 scores for yeast dataset, at different thresholds for coverage and sample count	220
C.8	Overlap percentage heatmap for yeast dataset at 98% coverage .	220
D.1	Distribution of F1 scores for sugar/amino acid SVM model in <i>A. thaliana</i> after outlier removal	223

Chapter 1

Introduction

1.1 Motivation

Proteins are responsible for essential functions in every cell and organism [1]. Transmembrane transport proteins (transporters) specifically are in charge of carrying molecules across biological membranes [2]. While some small and non-ionized molecules can pass the cellular membrane on their own through passive diffusion, many others, including ions, various small metabolites, essential nutrients such as sugars and amino acids, molecules involved in cell communication and signaling, and even entire proteins, have to rely on membrane transporters as a means to enter or exit the cell, or one of its intracellular compartments [3]. Studying the transporters that are present in a cell or a micro-organism, along with their transported substrates, is therefore an important task that provides crucial information to many scientific fields.

Membrane transporters can serve as a means for drug compounds to enter the cell and reach their destined targets in the intracellular environment [4]. Multi-drug efflux transporters play a major role in the resistance against antibiotics in pathogenic microorganisms [5], and in the acquired multi-drug resistance of cancer cells during chemotherapy [6]. Furthermore, it is estimated that membrane proteins in general make up over 60% of all drug targets [7].

The transporters that are present in the cellular membrane of an organism, along with the flux of each transported substrate, can give insights into how the intracellular networks interact with their environment [8]. The resulting computer models can then be used for phenotype prediction, metabolic engineering and drug design, among other applications [9][10].

Genetically engineered mammalian cells and micro-organisms are being used to synthesize many different products at an industrial scale, including biofuels, therapeutic proteins, animal feed, enzymes, and drug compounds [11][12][13]. Knowledge about the transporters that are located in (or could be added to) the respective cellular membranes can be used to tailor the nutrient concentrations in the growth medium towards the individual needs of the organism or cell [14][15], to prevent the export of intermediary metabolites in order to increase

the yield of subsequent pathways that lead towards the desired product [16], and to mediate the export of the synthesized molecule from the cell, which can help in simplifying the subsequent purification steps [17].

Many new organisms and strains are being discovered, and large parts of their proteomes have still not been functionally characterized [18]. This is also the case for membrane transporters, which continue to be among the most underrepresented groups of proteins in terms of available functional annotations [19][20]. However, experiments for determining transporter substrates are costly and labor-intensive. Computational methods can help in narrowing down the search space, e.g. by eliminating certain substrates from being considered for screening, or by providing ranked lists that can tell the researchers which substrate types (e.g. sugar, amino acid, potassium) to test for first.

Proteins are chains of amino acids that are held together by peptide bonds. These amino acid sequences are then folded into three-dimensional structures, and moved to the location in the cell where they carry out their function. In nature, there are 20 different types of amino acids that are commonly found in these protein sequences. The biological functions, the three-dimensional structure [21] and often also the destined intracellular location [22][23] of the protein are encoded in the order and length of its amino acid sequence.

A common method for predicting the unknown function of a protein is to search sequence databases for similar proteins, with software such as BLAST [24] and HMMER [25]. The goal is to find proteins with a high sequence similarity for which the desired annotation has already been experimentally determined. This information is then transferred to the new protein, under the assumption that sequence similarity is proportional to functional similarity. However, this approach also has its challenges [26]. On one hand, there are still many proteins for which the function has not yet been determined, and where no homologous proteins with known function can be found in available databases [18]. On the other hand, there are clusters of proteins that show high similarity in terms of their three-dimensional structure and/or their binding pockets, but simultaneously a low sequence identity to even closely related proteins, meaning that comparing their entire sequences might not be a suitable method for annotating their functions [27][28][29].

Most transporters are specific towards a particular substrate, or a group of substrates. Even multi-drug transporters with ten or more known substrates often have binding pockets that contain multiple, sometimes overlapping binding sites, which are capable of excluding molecules that are highly similar to those they transport [30][31][32]. This specificity is typically determined by a small fraction of the amino acids in the sequence, while the remainder of the protein shows a lower sequence conservation [28][33][34]. Some conserved sequence positions are directly related to the substrate selection of the protein, as they bind to specific atoms of transported molecule in the binding pocket [29][34], but others can also be necessary for achieving a particular structural fold, as seen in the selectivity filters of ion channels [35][36], or to carry out a mechanism that is shared by members of a protein family, e.g. the salt bridges that form or break in some carrier proteins, in order to change the protein structure in a way that releases the bound substrate on the other side of the membrane [29]. While sequence homology search is suitable for finding phylogenetically related transport proteins

of high sequence identity [37], it can therefore be a less reliable method for the automatic annotation of their transported substrates [38], as transporters of the same molecule can have different folds, low sequence identity and no common motifs, whereas transport proteins with high similarity in structure or sequence can be selective towards different substrates [27][28].

As an alternative to inferring functional information through sequence homology, supervised machine learning algorithms trained on encoded protein sequences have been used for the prediction of functional annotations [39][40][41]. The goal of this thesis is to expand on these approaches by creating a machine learning pipeline that can accurately predict membrane transporter substrates across a wide range of organisms and protein families. This will be divided into three parts: First, we will find suitable protein encodings and machine learning algorithms, along with filtering methods that only keep the useful data points for each sample. Second, we implement a more sophisticated method for creating membrane transporter datasets from published experimental data, along with a data-driven clustering method that can find a meaningful and well-separable subset of functional annotations for a set of proteins. Finally, we will further improve on the predictive performance of protein encodings and classification algorithms by incorporating recent advances in deep learning, natural language processing and protein structure prediction [21].

1.2 Objectives and Outline

The background information that is necessary for understanding the following chapters is summarized in Chapter 2. The first half starts with an introduction to proteins and membrane biology, followed by an overview of the different transport mechanisms found in membrane transport proteins. The focus of this section lies on how exactly the different types of proteins and protein complexes are able to select for their specific substrate molecules(s), both on a macro-molecular- and on an atomic scale. Specifically, we are interested in how sequence and structure enable the substrate transport function, so that we can rely on this information when we create protein datasets, feature encoding algorithms, and supervised machine learning pipelines. In Section 2.2, we start by introducing relevant software tools, websites and databases. Next, we build the mathematical background for statistical testing, correlation analysis and for the machine learning methods we used. We focus on Support Vector Machines (SVMs) and explain their internal workings in detail, along with different methods for extending and modifying the SVM algorithm for our purposes. Finally, the chapter ends with the evaluation metrics that were used throughout this thesis.

Chapter 3 then starts with a proof-of-concept machine learning model trained on two classes of carrier proteins in *A. thaliana*, which we analyze and optimize extensively. To this end, we test and compare various machine learning algorithms and protein feature encodings, and visualize the data with PCA, heatmaps and hierarchical clustering. Using the best-performing model as a starting point, we then apply the pipeline to additional organisms, and further adjust the model to address the resulting challenges and edge-cases. We try different approaches for increasing the number of samples available for training, such as training a model on one organism and applying it to another, combining the transporters from multiple species into a meta-organism dataset, and implementing a custom transformer that selects the best combination of feature encodings for a given set of substrate annotations. Finally, we also test a multi-class model with four substrates, where binary classifiers are trained in a one-vs-rest scheme. The code was released as a Python package that allows the user to automatically create training datasets and machine learning models for any set of substrate classes.

The first goal of Chapter 4 was to put more work into the dataset creation methods, and switch to a more accurate and more complex hierarchy of functional annotations. We developed a pipeline that automatically finds, filters and assembles the information available about membrane transporters in published literature, such as their sequences, motifs and known substrate molecules, as well as other related functional annotations. The dataset pipeline can be adjusted with several parameters, in order to find an appropriate trade-off between quantity and quality. Previously, we had also discovered that some sets of substrate classes result in more accurate models than others. Therefore, the second task was to employ statistical testing and correlation analysis, in order to find potential causes of these differences, and develop metrics that can predict good classification performance. To this end, we implemented a wide range of similarity measures between functional annotations, including chemical similarity between transported substrates, semantic similarity between GO terms, sequence

similarity between the annotated proteins, as well as similarity metrics derived directly from the GO graph, e.g. the difference in node degree. Finally, we used our findings to develop a constraint-based greedy clustering algorithm that can find a set of well-separable classes for a dataset of membrane transporters.

In Chapter 5, we then compare the SVM models from Chapters 3 and 4 against a deep neural network model that we developed for this task. Additionally, we test DL-based feature encodings and compare them to our previously developed methods. We derive features from structural models predicted with AlphaFold 2 [21], by converting them to 3Di sequences that describe the orientations and interactions of the amino acids, with FoldSeek [42]. We also take advantage of recent developments in natural language processing, by extracting embeddings from amino acid sequences and folded protein structures with protein language models (pLMs). Evaluation and comparison of the models was carried out on carrier proteins from *A. thaliana*, and on three types of human ion channels.

The project in its entirety and the results of the studies are summarized in the first section of Chapter 6. This is followed by an outlook section, where the potential next steps are discussed.

1.2.1 First author publications included in this thesis

Denger, A., & Helms, V. (2022). “**Optimized data set and feature construction for substrate prediction of membrane transporters**” *Journal of Chemical Information and Modeling*, 62(23), 6242-6257.

Abstract: Alpha-helical transmembrane proteins termed membrane transporters mediate the passage of small hydrophilic substrate molecules across biological lipid bilayer membranes. Annotating the specific substrates of the dozens to hundreds of individual transporters of an organism is an important task. In the past, machine learning classifiers have been successfully trained on pan-organism datasets to predict putative substrates of transporters. Here, we critically examine the selection of an optimal data set of protein sequence features for the classification task.

We focus on membrane transporters of the three model organisms *E. coli*, *A. thaliana*, *S. cerevisiae*, as well as human. We show that organism-specific classifiers can be robustly trained if at least 20 samples are available for each substrate class. If information from position-specific scoring matrices is included, such classifiers have F1 scores between 0.85 and 1.00. For the largest dataset (*A. thaliana*), a 4-class classifier yielded an F-score of 0.97. On a pan-organism dataset composed of transporters of all four organisms, amino acid and sugar transporters were predicted with an F1 score of 0.91.

Denger, A., & Helms, V. (2024). “**Identifying Optimal Substrate Classes of Membrane Transporters**” *PloS one*, 19(12), e0315330.

Abstract: Membrane transporters are responsible for transporting a wide variety of molecules across biological membranes, and are therefore involved in many important pathways in all organisms. However, experiments to identify the substrates of the vast number of newly sequenced gene products are costly and time-consuming. Machine learning models trained on protein data have successfully been used for assigning transporters into substrate classes, but so far, no method has been presented to find an optimal subset of functional annotations for transporter classification. Here, we implemented a pipeline for creating a dataset of transmembrane transporters, calculating pairwise similarity- and machine learning evaluation scores between the different classes of transporters, and annotating the dataset with a subset of Gene Ontology (GO) functional annotations that shows low redundancy and high pairwise machine learning scores. Applying correlation analysis and statistical tests, we found that the protein sets annotated to pairs of functional annotations should not be too similar or too dissimilar to each other in terms of sequence identity, and should have minimal annotation overlap. For a dataset containing 98.1% of transporters from *S. cerevisiae*, the pipeline automatically reduced the number of functional annotations from 287 to 11, while the GO terms had a median pairwise F1 score of 0.87 ± 0.16 . For a meta-organism dataset containing 96% of all transport proteins from *S. cerevisiae*, *A. thaliana*, *E. coli* and human, the number of classes was reduced from 695 to 49, with a median F1 score of 0.92 ± 0.10 between pairs of GO terms. When lowering the percentage of covered proteins down to 67%, the pipeline found a subset of 30 GO terms with a median F1 score of 0.95 ± 0.06 .

Denger, A., & Helms, V. (2025). “**Application of Protein Structure Encodings and Sequence Embeddings for Transporter Substrate Prediction**”, *Molecules*, 30(15), 3226.

Abstract: Membrane transporters play a crucial role in any cell. Identifying the substrates they translocate across membranes is important for many fields of research, such as metabolomics, pharmacology, and biotechnology. In this study, we leverage recent advances in deep learning, such as amino acid sequence embeddings with protein language models (pLMs), highly accurate 3D structure predictions with AlphaFold 2, and structure-encoding 3Di sequences from FoldSeek, for predicting substrates of membrane transporters. We test new deep learning features derived from both sequence and structure, and compare them to the previously best-performing protein encodings, which were made up of amino acid k-mer frequencies and evolutionary information from PSSMs. Furthermore, we compare the performance of these features either using a previously developed SVM model, or with a regularized feedforward neural network (FNN). When evaluating these models on sugar and amino acid carriers in *A. thaliana*, as well as on three types of ion channels in human, we found that both the DL-based features and the FNN model led to a better and more consistent classification performance compared to previous methods. Direct encodings of 3D structures with Foldseek, as well as structural embeddings with ProstT5, matched the performance of state-of-the-art amino acid sequence embeddings calculated with the ProtT5-XL model when used as input for the FNN classifier.

1.2.2 Coauthor publications during doctoral studies

Yang, W, Denger, A, Diener, C, Küppers, F, Soriano-Baguet, L, Schäfer, G, Yanamandra, AK, Zhao, R, rck, A, Schwarz, EC, Hart, M, Lammert, F, Roma, LP, Brenner, D, Christidis, G, Helms, V, Meese, E, Hoth, M & Qu, B (2022). “**Unspecific CTL Killing Is Enhanced by High Glucose via TNF-Related Apoptosis-Inducing Ligand.**” *Frontiers in immunology* 13 831680

Abstract: TNF-related apoptosis inducing ligand (TRAIL) is expressed on cytotoxic T lymphocytes (CTLs) and TRAIL is linked to progression of diabetes. However, the impact of high glucose on TRAIL expression and its related killing function in CTLs still remains largely elusive. Here, we report that TRAIL is substantially up-regulated in CTLs in environments with high glucose (HG) both *in vitro* and *in vivo*. Non-mitochondrial reactive oxygen species, NF κ B and PI3K/Akt are essential in HG-induced TRAIL upregulation in CTLs. TRAIL^{high} CTLs induce apoptosis of pancreatic beta cell line 1.4E7. Treatment with metformin and vitamin D reduces HG-enhanced expression of TRAIL in CTLs and coherently protects 1.4E7 cells from TRAIL-mediated apoptosis. Our work suggests that HG-induced TRAIL^{high} CTLs might contribute to the destruction of pancreatic beta cells in a hyperglycemia condition.

My contribution: I performed the differential expression analysis of Agilent microarray data with the Linear Models for Microarray Data (limma) R package [43], including background correction with the NormExp method, quantile normalization, and multiple filtering steps for improving data quality. Pre-

processing was carried out with R packages from tidyverse [44] and Bioconductor [45]. Using limma, a linear model was calculated between samples in high glucose (HG) or normal glucose (NG) conditions, and differentially expressed genes were identified with a t-test. The associated p-values were calculated for each gene with an empirical Bayes method and adjusted for multiple testing with the Benjamini-Hochberg method [46]. Among the DE genes, I performed gene annotation enrichment analysis to find Gene Ontology (GO) terms [47] and KEGG pathways [48] that are over-represented in one of the gene sets. Additionally, I carried out another pathway enrichment analysis based on protein interaction subnetworks, using the pathfindR package [49]. I submitted the gene expression datasets to ArrayExpress (accession E-MTAB-11441), after applying the required data processing steps [50]. Finally, I visualized results of the differential expression analysis and gene annotation enrichment analyses with the seaborn [51] and matplotlib [52] packages in Python, and with the ggplot2 [53] and gplots [54] packages in R.

Cappello, S, Sung, HM, Ickes, C, Gibhardt, CS, Vultur, A, Bhat, H, Hu, Z, Brafford, P, Denger, A, Stejerean-Todoran, I, Köhn, RM, Lorenz, V, Künzel, N, Salinas, G, Stanisiz, H, Legler, T, Rehling, P, Schön, MP, Lang, KS, Helms, V, Herlyn, M, Hoth, M, Kummerow, C, Bogeski, I (2021). **“Protein Signatures of NK Cell-Mediated Melanoma Killing Predict Response to Immunotherapies.”** *Cancer Research*, 81, 21:5540-5554.

Abstract: Despite impressive advances in melanoma-directed immunotherapies, resistance is common and many patients still succumb to metastatic disease. In this context, harnessing natural killer (NK) cells, which have thus far been sidelined in the development of melanoma immunotherapy, could provide therapeutic benefits for cancer treatment. To identify molecular determinants of NK cell-mediated melanoma killing (*NK_mK*), we quantified NK-cell cytotoxicity against a panel of genetically diverse melanoma cell lines and observed highly heterogeneous susceptibility. Melanoma protein microarrays revealed a correlation between *NK_mK* and the abundance and activity of a subset of proteins, including several metabolic factors. Oxidative phosphorylation, measured by oxygen consumption rate, negatively correlated with melanoma cell sensitivity toward *NK_mK*, and proteins involved in mitochondrial metabolism and epithelial-mesenchymal transition were confirmed to regulate *NK_mK*. Two- and three-dimensional killing assays and melanoma xenografts established that the PI3K/AKT/mTOR signaling axis controls *NK_mK* via regulation of NK cell-relevant surface proteins. A “protein-killing-signature” based on the protein analysis predicted *NK_mK* of additional melanoma cell lines and the response of patients with melanoma to anti-PD-1 checkpoint therapy. Collectively, these findings identify novel NK cell-related prognostic biomarkers and may contribute to improved and personalized melanoma-directed immunotherapies.

My contribution: I performed a correlation analysis to find genes in an external gene expression dataset from Hugo et al. [55] that correlate with the NK92 killing signature, then used those genes to train a Support Vector Machine (SVM) model that can predict melanoma patient sensitivity to anti-PD-1

melanoma immunotherapy, by categorizing patients into responders and non-responders, with relatively high accuracy (81% sensitivity and 78% specificity in a 5-fold cross validation). Using a χ^2 test, the 17 top-ranking genes in terms of predictive value were selected for this model. I visualized the SVM hyperplane and training samples as a 2D scatter plot, after performing a principal component analysis (PCA) to reduce the number of feature dimensions from 17 to two. Training the model on only the first two principal components improved the evaluation further (sensitivity and specificity of 86% and 85%, respectively). Finally, I applied a gene-wise Z-score to the genes that led to the best-performing model, applied Ward's hierarchical clustering, and visualized the results as a clustered heatmap with a dendrogram, using the seaborn [51] Python package. The scikit-learn [56] Python package was used for training and evaluation of the SVM, for the feature selection, and for performing the PCA. Pre-processing of labels and features for the machine learning training dataset was carried out with the tidyverse framework [44] in R.

Chapter 2

Background

2.1 Biology

In this section, an overview is provided of the major protein families and transport mechanisms that exist across the tree of life, while paying special attention to how the amino acid sequence, the three-dimensional fold, and the physicochemical properties of the individual amino acids influence the substrate selection and general function. The goal is to not just get an comprehensive understanding of transmembrane transport proteins, but to also look for similarities and patterns that machine learning models could take advantage of when trying to predict the substrate selectivity or the transport mechanism of newly discovered and/or functionally uncharacterized proteins. These conclusions will be discussed in a short summary at the end of the section.

2.1.1 Proteins

The functions of genes in the cell are carried out by their gene products. Genes are transcribed to RNA molecules, which in some cases can carry out tasks directly, but in most cases are further translated into proteins. They are responsible for many important functions in the cell: Enzymes catalyze chemical reactions that break down nutrients and process subsequent metabolites, structural proteins make up the *cytoskeleton* that gives rigidity to cells and their cellular compartments, transcription factors regulate the rate at which genes are transcribed, and transporters carry molecules from one location to another [57].

At a molecular level, proteins are chains of amino acids, held together by peptide bonds. The individual amino acids are made up of an amino-group ($-\text{NH}_2$), a carboxy-group ($-\text{COOH}$) and a *residue* group ($-\text{R}$), all held together by a central carbon atom called C_α (see Figure 2.1). While the former two components, together with C_α , make up the *backbone* of the protein, the residue determines the name of the amino acid, as well as its size, flexibility and its physicochemical properties [58]. Amino acids can form *peptide bonds* between each other; a hydroxy group ($-\text{OH}$) from the carboxy group of one amino acid, and a hydrogen atom from the amino group of another amino acid, form a water molecule (H_2O),

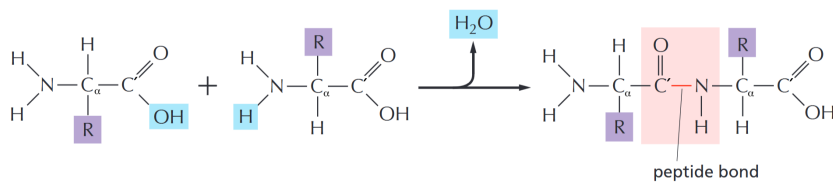


Figure 2.1: Two proteinogenic amino acids, each made up of an amino group (NH_2), a side chain residue R , a carboxy group (COOH) and a central C_α atom, form a peptide bond through a condensation reaction. Adapted from Alberts et al. [57].

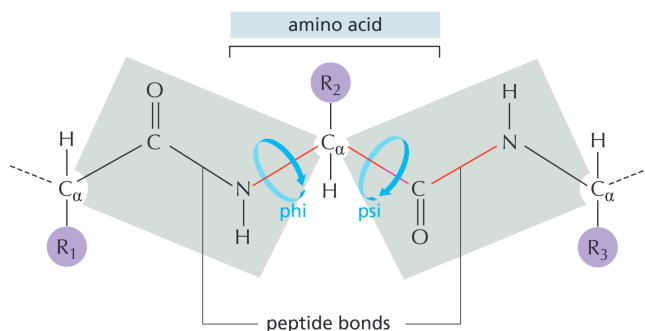


Figure 2.2: The flexibility of the protein backbone is achieved by rotation around the bonds formed by the central C_α atom, and limited by interactions between residues and backbone atoms. The peptide bond shows little flexibility, it is at the center of the so-called *peptide plane*, shown in grey. Figure from Alberts et al. [57].

allowing the carbon to bind directly to the nitrogen (see Figure 2.1). Chains of fewer than 20 amino acids are typically called *peptides*, whereas longer chains are referred to as proteins [58]. In total, the standard genetic code encodes 20 different types of these *proteinogenic* amino acids (see Figure A.1), and two more can be added by special translation mechanisms (i.e. selenocysteine and pyrrolysine) [57][59].

Each amino acid in the chain is equipped with one or two axes of rotation around the bonds formed by the central C_α atom, which gives the protein backbone some flexibility (see Figure 2.2). The range of possible angles depends on the residue of the amino acid; proline is a special case where the first angle is completely restricted, as the side chain is covalently bound to the backbone and forms a ring, while glycine is the most flexible amino acid due to its small side chain (see Figure A.1). The peptide bond shows little movement [58]. Due to this flexibility, the amino acids in the protein are capable of forming covalent and non-covalent bonds with each other, which allows for the protein to be folded into three-dimensional (3D) shapes. At the lowest level, hydrogen bonds in the backbone build *secondary structures*, such as α -helices and β -sheets (see Figure 2.3). In helices, the backbone oxygen of the n th amino acid in the sequence forms a hydrogen bond with the backbone amino group (NH) of the $(n + k)$ th amino acid [60][61]. By far the most common helical structure is the α -helix ($k = 4$) (see Figure 2.3a), although π -helices and 3_{10} -helices ($k = 3$ and $k = 5$, respectively) have also been observed in nature [62][63]. The other common secondary structure is the β -sheet, where hydrogen bonds are formed between the backbones of two or more parallel (or anti-parallel) strands of the

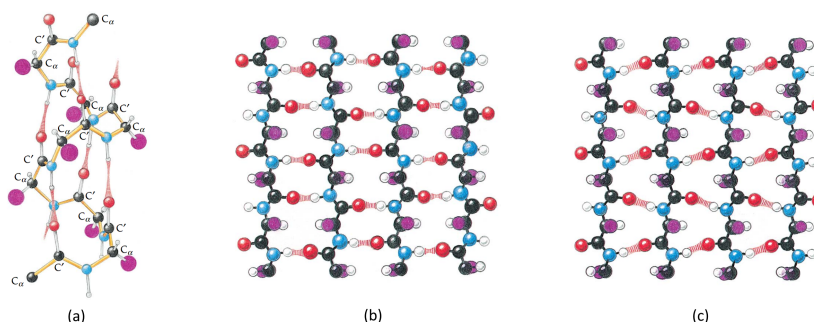


Figure 2.3: Three common types of secondary structures found in proteins. Spheres represent atoms that are colored according to their element (black=carbon, blue=nitrogen, red=oxygen, white=hydrogen). Red dotted lines are hydrogen bonds, continuous lines represent covalent bonds. (a) This figure shows an α -helix, which is held together by backbone hydrogen bonds between the n th and $n+4$ th amino acid in the sequence. (b) An anti-parallel β -sheet, formed through hydrogen bonds between four different parts of the protein sequence. (c) A parallel β -sheet. Drawings are from Branden and Tooze [58].

protein sequence (see Figure 2.3b,c) [60]. The remaining parts of the protein sequences are typically referred to as turns, loops or coils; they can be more or less flexible depending on the included amino acids and their interactions with other residues in the 3D structure [58][64].

From interactions between these secondary structures, the tertiary structure of the protein is formed. Whereas the secondary structures rely solely on hydrogen bonds between atoms in the backbone of the proteins, the protein folding process also utilizes bonds between two residues, or between a residue and a backbone atom. These include hydrogen bonds, hydrophobic interactions, covalent disulfur bonds between two cysteines, van der Waals interactions, and ionic bonds between a basic and an acidic residue [58][65]. The presence of a residue with desired properties at a particular position in the protein can therefore be very important for the shape and function of the protein [66]. For this reason, these crucial positions often remain *conserved* during evolution [67], although a replacement with a chemically similar residue can occur as well [68]. In turn, it is possible to find the sequence positions with highest functional importance by comparing different variants of a protein across multiple species, and finding those positions that remain constant in terms of size, steric effects, charge, hydrophobicity, and other physicochemical properties [69].

If a sub-sequence of three or more amino acids is conserved then it is referred to as a *sequence motif* [70], while a conserved part of a 3D-structure is called a *protein domain* [58]. Protein families are sets of homologous proteins that typically share at least one domain or motif [71][72]. One type of protein domain that is found in enzymes, receptors and membrane transporters is the *active site*, or *binding pocket* of the protein. The residues that protrude into this cavity have chemical properties that mirror those of its substrate, ensuring that the molecule docks to the protein in the desired orientation, which changes the chemical properties of the substrate and the protein in a particular way [58]. In enzymes, for example, some of the valence electrons of the bound compound can be pulled towards polar residues in the active site, which lowers the energy activation barrier for (i.e. which *catalyzes*) a chemical reaction on the molecule

[73]. In certain types of membrane transport proteins, the binding of a substrate can lead to a series of conformation changes in the three-dimensional structure of the protein that cause it to close on one side of the membrane and open on the other, where the substrate is then released again (see Section 2.1.3.2, Figure 2.12b) [28].

These binding pockets are typically substrate-specific, meaning that the molecule, or a set of similar molecules, fit into the protein just like a key fits into a key-hole (see Section 2.1.3.2, Figure 2.11b) [73], although larger cavities with binding sites for multiple different compounds exist as well (see Sections 2.1.3.1 and 2.1.3.2) [74]. Many pharmaceutical compounds are tailor-made to fit into the binding pocket of their respective target proteins, in order to interfere with their functions in some way [75]. Finding molecules that interact with the binding pocket of an enzyme or a membrane transporter is therefore a topic of ongoing research [76].

Another type of structural domain is the protein-protein binding interface. Through these areas on their solvent-accessible surface, proteins are capable of binding to each other, either temporarily (e.g. to deliver a substrate, as shown in Figure 2.6), or because they carry out their functions as a part of a protein complex, which is also referred to as the *quaternary structure* of a protein [77][78]. Some protein complexes, called *homomers*, are made up of multiple copies of the same protein [79](see Figures 2.15b, 2.17a), while heteromers are made up of different proteins [80] (see Figures 2.6, 2.10). Homomers can also be assembled into larger complexes, as shown e.g. in Figure 2.8

2.1.2 Membranes

Biological membranes are dynamic barriers made up of lipids, carbohydrates and proteins that separate the intracellular environment from the outside (see Figure 2.4). Membranes can also occur inside the cell itself, where they surround intracellular compartments that are responsible for distinct functions [81][82].

2.1.2.1 The structure of the lipid bilayer membrane

The individual lipids are composed of a hydrophobic tail and a hydrophilic head, which are arranged in the form of a *lipid bilayer*, where the tails are pointing inward, while the hydrophilic heads interact with the aqueous solution that surrounds the membrane [83]. At sufficiently high temperatures, the lipid bilayer is in a liquid-crystalline phase, meaning that it exhibits properties of liquid and crystalline materials, and allows the individual molecules to move around the membrane laterally [81][84]. Below their critical melting temperature T_m , the membrane lipids can undergo a reversible phase transition into a gel, which has been shown to affect membrane permeability, among other things [85][86].

Membrane-bound carbohydrates are attached to the extracellular surface of the membrane, covalently bonded to so-called *glycoproteins* and *glycolipids* that are themselves integrated into the lipid bilayer [87]. This outer layer of polysaccharides helps to protect the cell from pathogens [88], regulates cell-cell interactions and signaling [89][90], and can act as anchor points for adhesion to other cells or other microorganisms through sugar-binding glycoproteins on the opposing extracellular surfaces [91].

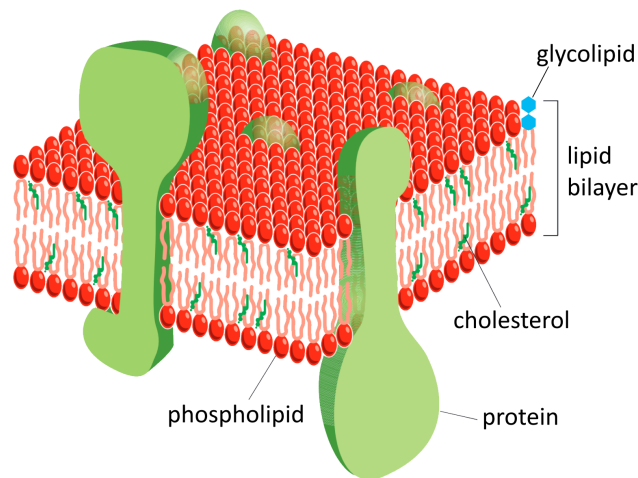


Figure 2.4: Schematic view of the lipid bilayer membrane. The hydrophilic heads of membrane lipids are facing the surrounding aqueous solution, while the hydrophobic/lipophilic tails interact with each other, forming the membrane environment. Hydrophobic domains in membrane proteins allow them to be integrated into the lipid bilayer. The concentration of sterols, such as cholesterol, regulates the flexibility and fluidity of the membrane. Glycolipids contain polysaccharide chains of varying length that reach into the watery solution around the membrane [57]. Adapted from Alberts et al. [57].

Proteins can be integrated into biological membranes, where they fulfill a wide range of functions [81]. These membrane proteins can provide structural support by anchoring to the cytoskeleton and the extracellular matrix [92], act as receptors that trigger cellular responses when detecting external signals [93], and are capable of carrying ions and molecules across the membrane barrier [2]. On one hand there are *peripheral* membrane proteins, which are attached to the surface of the membrane through covalent or non-covalent bonds [94]. GPI-anchored proteins, for example, are covalently bound to a membrane glycolipid, and this *lipid anchor* is then integrated into the fluid-crystalline structure where it can move around laterally (see Figure 2.5f) [95]. Other peripheral membrane proteins are covalently bound to a membrane lipid (Figure 2.5e), or contain a single, lipophilic transmembrane helix (TMH) that reaches into the lipid bilayer, while the rest of the protein structure remains in aqueous solution outside of the membrane (Figure 2.5d) [94]. Also, some proteins can, temporarily or permanently, bind to the membrane through non-covalent interactions, as is the case with the enzyme cytochrome *c* in the mitochondrion [96] or the cytoskeletal protein spectrin [97]. *Integral* membrane proteins, on the other hand, reach all the way through the membrane. They contain a larger number of outward-facing hydrophobic residues that allows large parts of the protein to remain in the hydrophobic membrane environment. These residues are often part of secondary structures, such as transmembrane helices (TMHs), and more rarely as transmembrane β -sheets [98]. Amino acid side chains facing the aqueous solution around the membrane are more hydrophilic, which allows the protein to maintain its shape and orientation in the membrane [99]. Whereas single-pass integral membrane proteins only traverse the membrane once (Figure 2.5a), multiple TMHs (Figure 2.5a) or β -strands (Figure 2.5b) in multi-pass proteins

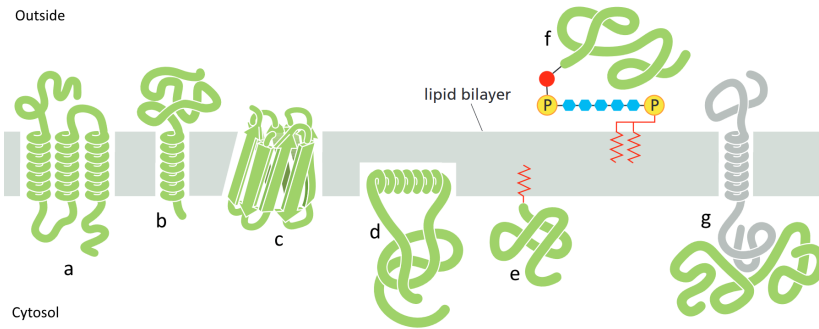


Figure 2.5: Different ways in which proteins can be attached to or integrated into the lipid bilayer membrane. (a) Multi-pass transmembrane helix (TMH) protein. Can contain a central pore or binding pocket when the TMHs form a helical bundle structure with each other, or a complex with another multi-pass TMH protein. (b) Single-pass TMH protein with extracellular globular domain. (c) Membrane protein with β -barrel super-secondary structure. (d) Peripheral membrane protein with α -helix. (e) Peripheral protein covalently bonded to membrane lipid. (f) Membrane protein attached to the lipid bilayer through a GPI anchor. (g) Heteromeric protein complex of a single-pass integral transmembrane protein, and a peripheral protein attached to its binding interface. Adapted from Alberts et al. [57].

can form *super-secondary structures* by interacting with each other inside the membrane environment [58]. Examples of these are α -helical bundles (see Figure 2.11a) and β -barrels that form structurally stable pores in the membrane, and are therefore commonly found in transmembrane transporters [61]. Finally, membrane proteins can be part of protein complexes, and form heterodimeric or homodimeric quaternary structures with other integral or peripheral membrane proteins (Figure 2.5g) [79][100][101][34].

2.1.2.2 Different types of membranes

Both the cell itself and its intracellular compartments are encased by membranes. Some organelles and microorganisms are surrounded by both an outer and an inner membrane, separated by an inter-membrane space. While all of these membranes share the same lipid bilayer structure, they differ in their molecular composition, according to the functions they are responsible for [83][102]. While most tasks are carried out by proteins, the lipid composition, (i.e. the proportional amounts of different phospholipids, glycolipids and sterols) of a membrane can influence and regulate the activity of integrated and peripheral proteins [83][29][57].

The plasma membrane surrounds the cell, and therefore specializes in the selective exchange of molecules with the extracellular environment, as well as signal transduction [102]. Furthermore, it serves a structural function, as it contains proteins that bind to the cytoskeleton, and help the cell maintain its shape [97].

The dual-membrane envelope that surrounds the nucleus in eukaryotes contains so-called *nuclear pore complexes* (NPCs) spanning both membranes. NPCs facilitate the export of mRNA and other molecules involved in translation (i.e. protein synthesis) and the import of molecules involved in gene regulation and transcription, among other tasks [103]. After the mRNA exits the NPCs, it arrives in flattened membrane vesicles referred to as *cisterna*. They make up the

endoplasmatic reticulum (ER), which is continuous with the outer membrane of the nuclear envelope. Ribosomes, i.e. the protein-RNA complexes that translate mRNA into protein sequences, can attach to the ER membrane through non-covalent interactions, in which case they are referred to as ER-bound ribosomes (as opposed to free or cytosolic ribosomes) [104].

Mitochondria in eukaryotic cells are surrounded by two membranes as well. The outer membrane contains a considerable number of β -barrel porins (see Section 2.1.3.4) [105], whereas the inner mitochondrial membrane, along with the inter-membrane space, hold the integral and peripheral membrane proteins that make up the electron transport chain. Together, they enable the synthesis of adenosine triphosphate (ATP), a molecule that transfers energy to a wide range of crucial cellular processes [106][96]. The inner lipid bilayer is folded in a wave-like fashion, giving it a larger surface area compared to the outer mitochondrial membrane [107].

The presence of an outer membrane in addition to the regular plasma membrane is also the main difference that distinguishes gram-negative bacteria, such as *E. coli*, from their gram-positive counterparts. Mitochondria (and chloroplasts) show similarity to gram-negative bacteria in this way, leading to the assumption of an evolutionary relationship, which is referred to as the endosymbiosis theory [108]. Both are encased by outer membranes lined with β -barrel porins, which exclusively occur in outer membranes [109], and the inter-membrane space of either contains membrane proteins and enzymes that interact with ions and metabolites [110][96][111]. In contrast to the eukaryotic organelles, gram-negative bacteria are surrounded by a pronounced layer of polysaccharides that is covalently bound to the outer membrane [112][113]. The inter-membrane space of the gram-negative bacterial envelope is filled with a third barrier consisting of long chains of sugars and amino acids, a substance that is referred to as *peptidoglycan*. The peptidoglycan layer is more pronounced in gram-positive bacteria, where it serves as a protective barrier for the organism, as it lacks an outer membrane [114].

2.1.2.3 Membrane transport without integral transport proteins

Membranes are semi-permeable, selective barriers that separate two aqueous environments by regulating the exchange of molecules between them. Some molecules can pass through the membrane on their own, driven by concentration gradients. The highest diffusion rates are observed for non-polar gases such as oxygen and carbon dioxide. Small and *polar* molecules like ethanol and water can pass the membrane barrier through passive diffusion as well, but at a much lower rate compared to their non-polar counterparts. Molecules that are comparatively larger than that, as well as ions and small molecules that carry an electric charge, diffuse across biological membranes either in very small quantities or not at all [3].

Membrane vesicles are spherical lipid bilayer structures that are used both for intracellular transport of molecules, as well as for their import and export through biological membranes. Vesicles are pinched off from one membrane, and can then merge with another membrane, in order to release their contents. The export of cellular (waste) products out of the cell through membrane vesicles is called

exocytosis [57]. Endocytosis, on the other hand, is divided into two categories: Phagocytosis allows a cell to engulf large particles, whereas pinocytosis takes in fluids and certain macromolecules [115]. Vesicles created by endocytosis are mainly transported to the endosome, an intracellular compartment where the molecules are sorted, and then metabolized or secreted again [116]. Another purpose of this system is to change the molecular composition of the membrane according to the current needs of the cell; large fractions of the cell membrane go through a continuous cycle of endocytosis and exocytosis [116]. For example, the presence of insulin at the corresponding receptor protein triggers the exocytosis of membrane vesicles containing transmembrane glucose transporters, which in turn increases the glucose uptake into the cell [117]. Protein activity and protein complex formation can be regulated through changes in membrane lipid composition [83]. Generally, the endo- and exocytosis system is capable of transporting molecules and particles across the membrane that cannot easily pass this barrier through either passive diffusion or transmembrane transport proteins [3]. This process is mainly found in eukaryotes [57].

2.1.3 Transmembrane transport proteins

The remaining molecules are transported through the membrane barrier using integral transmembrane proteins, which will be discussed in the following section.

Transport mechanisms of membrane transport proteins can roughly be categorized into five classes according to the transporter classification system [118]: Primary active transporters, carrier proteins, channels and pores, group translocators, and electron transporters.

The 3D structures of transmembrane transport proteins can be roughly categorized into two groups: those that form a pore or binding pocket in the membrane using a bundle of transmembrane helices (TMHs), and barrel-shaped pores made up of β -strands (see Figures 2.5a and 2.5c, respectively). Some transporters act on their own, some form homomers or heteromers in the membrane, and some are part of larger complexes that can contain non-transporter proteins, such as enzymes [119][28][57].

2.1.3.1 Primary active transporters

Primary active transport moves molecules across cellular membranes against a their concentration gradient, using a primary energy source. In most cases, the energy stems from the hydrolysis of adenosine tri-phosphate (ATP) to adenosine di-phosphate (ADP). Turning ADP into ATP by adding a phosphate allows the cell to store high amounts of energy in the molecule, with the goal of delivering that energy to a place in the cell where it is needed, where it is then released again by hydrolyzing ATP back to ADP (akin to a rechargeable battery) [57]. There are primary active transporter families that use other direct energy sources, such as light absorption [120] or redox reactions [121].

One prominent family of primary active transporters that is found across all kingdoms of life is that of the ATP-binding cassette (ABC) complexes [122]. There are dozens of ABC transporter sub-families for carrying a wide variety of molecules, including ions, small organic compounds, peptides, polysaccharides

and xenobiotics [119]. ABC complexes are always made up of two α -helical bundle transmembrane domains (TMD) that bind the substrate, and two nucleotide binding domains (NBD) that interact with ATP. Sub-families can contain additional domains [34]. After successful binding of both ATP and the substrate to their respective domains, the hydrolysis reaction of ATP to ADP leads to a conformational change in the protein structure, which opens the binding pocket on the other side of the membrane and allows the substrate to be released [123]. The exact sequence of events can differ depending on the sub-family [34]. In prokaryotes, the four canonical domains of ABC transporters are typically made up of four separate proteins, and both importers and exporters are commonly found. In eukaryotes, on the other hand, the four domains are present either in one long protein sequence, or the complex is a heterodimer or homodimer where each protein contains two domains, and is referred to as an ABC *half-transporter* [122][124].

ABC transport complexes can be classified into seven groups according to their structures, substrates, and mechanisms [125][34].

Types I-III are prokaryotic importers. Type I complexes mainly import sugars, amino acids, and peptides, while types II-III pump micro-nutrients into the cell, such as folate or cobalamin (Vitamins B9 and B12, respectively). Transporters of Type I and II employ extracellular or, in case of gram-negative bacteria, periplasmic substrate binding proteins (SBPs), which bind to the transported molecule first. Extracellular SBPs are more likely to be permanently attached to the protein complex, while periplasmic SBPs typically float freely in surrounding aqueous environment, and bind temporarily. Some SBPs can bind multiple different substrates, and some ABC transporters are compatible with more than one SBP, meaning that substrate selection for this type of transport mechanism is, at least in part, carried out by proteins that are not integrated into the membrane [126][125]. Type II ABC transporters do not possess special binding pockets internally, instead their interior is lined with hydrophobic residues that act like a smooth and slippery surface [127]. On the other hand, their SBPs tend to show a higher substrate specificity compared to those of Type I ABC complexes [125][34].

An example for a Type I ABC complex is the MalFGK₂ transporter in the inner membrane of *E. coli* (see Figure 2.6). It is made up of two copies of MalK that provide the NBD domains, while MalF and MalG are the two TMDs, and MBP acts as the substrate binding protein (SBP). MBP travels freely in the periplasm, where it binds to the substrate, and subsequently docks to the complex. Next, the binding of ATP to the NBDs leads to a conformational change that transfers the molecule into the binding pocket and removes the SBP from the complex. Inside the cavity, the MalF protein forms a binding pocket for maltose, where the residues of Asp437, Asp440, Asp376, Ser329 and Tyr325 are situated in the correct positions to form hydrogen bonds with oxygens and hydroxyl groups in the bound molecule. Finally, after ATP hydrolysis takes place in the NBDs, the transmembrane complex opens the other side, where the substrate can be released into the cell [128][34].

Energy-coupling factor (ECF) transporters, also referred to as Type III ABC transporters, do not employ external SBPs for substrate binding. Instead, one of their two TMD domains, called the *S-component*, is solely responsible for

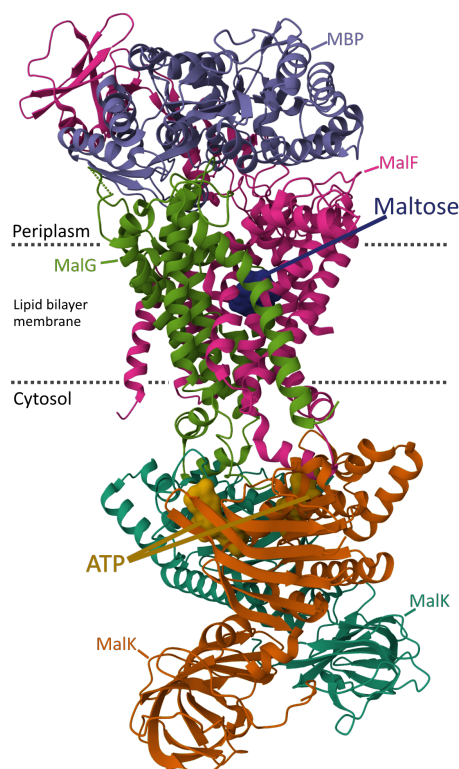


Figure 2.6: The 3D structure of the maltose-importing ABC complex MalFGK_2 from *E. coli*, with maltose and two ATP molecules located in their respective binding pockets. The complex contains two identical NBDs in the cytoplasm (MalK), two different TMDs made up from TMHs (MalG , MalF), one of which reaches into the periplasm, where it helps in temporarily connecting to the substrate binding protein MBP. Structure from Oldham et al. [128] and rendered with Mol* [129] (PDB 2R6G), inspired by Figure 2 from Thomas and Tamp   [34].

binding the substrate. The S-component is always made up of six TMHs and shows high structural conservation, while the fold of the other TMD is more variable. The two TMDs found in ECF complexes are often structurally dissimilar, which is unusual for ABC transporters. Both TMDs show low levels of sequence conservation [130]. As in type I and type II complexes, the binding of ATP to the NBDs opens the binding pocket towards the extracellular environment [131]. The transported substrate of an ECF complex can typically be inferred from its S-component, as they tend to be specific towards a certain substrate (e.g. BioY for Biotin, CbiM for Cobalt etc., see Table 1 in Rempel et al. [132] for all known S-components) [132].

Types IV and V occur in both prokaryotes and eukaryotes. Most complexes of these types are exporters, although there are some exceptions. They are capable of carrying a very wide range of substrates, including xenobiotics, lipids, peptides, ions, hormones and cholesterol [34]. Many ABC complexes that belong to these two types are multi-drug resistance (MDR) transporters, meaning they carry various drugs and xenobiotics, and are implicated both in antibiotic resistance by pathogenic microorganisms, and in drug resistance of can-

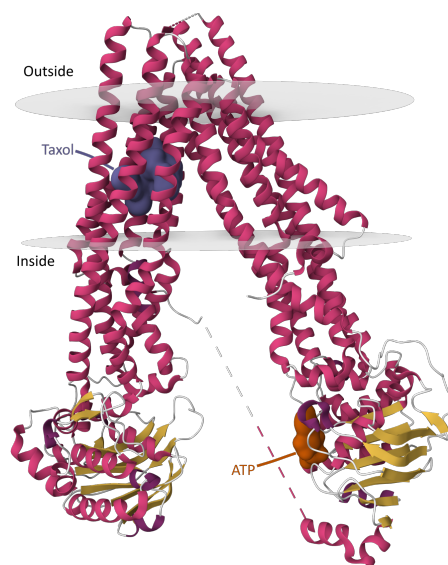


Figure 2.7: The human multi-drug resistance protein ABCB1 (MDR1, P-glycoprotein), with the cancer drug Taxol bound to its large cavity, and ATP bound to one of the NBDs. ABCB1 is a Typ IV ABC complex, and a drug efflux pump involved in the resistance of cancer cells to pharmacotherapy [133]. The entire protein is encoded in one continuous amino acid sequence, and was colored according to secondary structures (helix = red/purple, β -sheet = yellow). The dotted line in the middle represents a linker sequence of around 60 amino acids that shows little to no secondary structure formation [134]. Source: Kurre et al. [32] (PDB:9CTF), rendered with Mol* [129]

cer cells during chemotherapy [135]. The three most prominent ABC drug exporters in human cells are *multidrug-resistance-associated protein 1* MRP1 (ABCC1), P-glycoprotein (MDR1, Pgp, ABCB1), and *breast cancer resistance protein* (BCRP, ABCG2) [34][133]. Substrates of MDR transporters tend to be lipophilic. In some cases, the drug compounds are already located inside the membrane environment, where they can then enter the binding pocket of the transport complex laterally, through membrane-facing gates. The drug can then be exported, or it is *flopped* [136]. A *floppase* is a protein complex that “flops” a lipophilic molecule, such as a drug, from the inner layer to the outer layer of the (bi-layer) membrane, while a *flippase* does the same thing in the reverse direction [133]. While ABC complexes that act as MDR efflux pumps can carry a wide range of substrates, they are still selective, and only carry a small subset of the molecules that are present in the surrounding environment. ABCB1, for example, contains at least four drug binding pockets/sites that are located in one large cavity formed by the TMDs, where residues from all twelve TMHs can be involved in substrate interactions (see Figure 2.7) Kurre et al. [32]. This, together with the fact that the complex can recognize small differences between molecules during its selection process, makes the identification of possible substrates, modulators and inhibitors of ABCB1 a challenging task for researchers [30][133]. Some MDR transporters, as well as structurally related and homologous ABC complexes like MsbA from *E. coli*, are also involved in the flipping and flopping of membrane lipids, from one layer of the membrane to the other [137][138]. While large cavities, like the one found in P-glycoprotein,

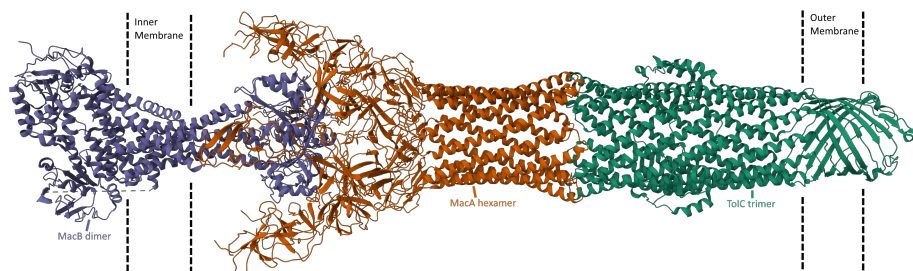


Figure 2.8: The structure of the MacAB–TolC tripartite multidrug efflux pump. MacB is a member of the ABC transporter complex family. It takes up drugs and xenobiotics in the periplasm, and pumps them out of the cell through the MacA and TolC complexes. The mechanical motion that is generated by the binding of ATP to the intracellular NBDs is transferred to MacA, causing it to close, and pump out the substrate, similarly to a bellows. ATP hydrolysis then reverts the movement [140]. Structure by Fitzpatrick et al. [141] (PDB:5NIK, rendered with Mol* [129]).

are typical for *type IV* ABC complexes, the pores of *type V* ABC transporters more closely resemble channels or slits. However, unlike regular ion channels (as discussed later in Section 2.1.3.5), they are typically hydrophobic on the inside, are capable of multi-drug recognition, and can take up substrates laterally from the membrane lipid environment [34]. Interestingly, the slit-like pore of Type V ABC complex ABCG2 (BCRP) leads to a preference for flat, polycyclic and hydrophobic compounds. The chemical properties of residues that line this opening, and the lipophilic environment around this opening that molecules often have to enter before being transported, also contribute substrate selection [139][80].

Types VI and VII again only occur in prokaryotes, and encompass *extractors* and *mechanotransmitters*, respectively [34]. Extractors, such as LptB₂FG, remove lipopolysaccharides (LPS) from the membrane they are in, meaning they do not actually transport a substrate from one side to the other. Instead, LptB₂FG is part of a larger protein complex that removes LPS from the inner membrane of gram-negative bacteria, transports them through the periplasm, and inserts them into the outer membrane [113]. An example for a mechanotransmitter is the MacB protein from *E. coli*. Again, this protein does not transport substrates through the inner membrane, where it is integrated. Rather, it is part of the MacAB–TolC tripartite multidrug efflux pump, which takes up drugs and xenobiotics from the periplasm, and pumps them through the outer membrane. This structure is made up of three parts (see Figure 2.8). MacB is a dimeric complex and is integrated into the inner membrane. It contains two intracellular NBDs that interact with ATP. In the periplasm, this structure is attached to six copies of MacA that are symmetrically arranged into a barrel-like structure, where each protein contributes two neighboring α -helices, as well as a tail of β -strands and loops, to the hexamer. This, in turn, connects to three copies of TolC that form a β -barrel, where each protein provides four neighboring β -strands, plus multiple α -helices that are attached to each strand [141]. The TMD of MacB lacks a binding pocket for substrates. Analysis of the 3D structure led to the conclusion that a change of the protein structure due to ATP hydrolysis carried out by the intracellular NBDs transfers mechanical movement through the inner membrane, leading in turn to conformational changes in the structure of the

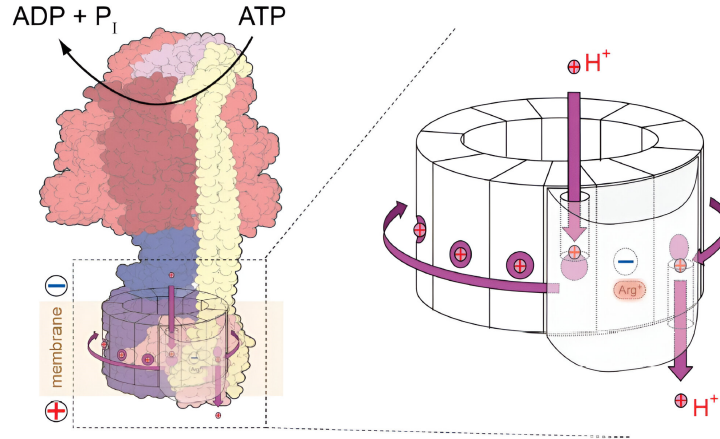


Figure 2.9: Schematic mechanism of a rotary ATPase that pumps protons against their electrochemical gradient, as described in Section 2.1.3.1. The enzyme complex (red) turns energy from ATP hydrolysis into mechanical motion of the stalk (blue), which reaches into the ATP hydrolysis domain (the overlap is colored in dark red), and is connected to a rotating barrel made up of transmembrane helices (purple). An additional integral membrane domain (yellow) is attached to the enzyme complex, and contains two half-channels that load protons into the negatively charged binding sites, and removes them again after the substrate has traveled around the circle, with a conserved arginine residue. Adapted from Zubareva et al. [144]

connected MacA complex. The substrate likely enters the complex through an opening on the periplasmic side of MacB, where it then passes a gating ring of six glutamines in MacA that might prevent backflow [141]. Binding ATP then connects the two NBDs to each other, and passes that movement on to MacA, which pushes the substrate out of the complex, similarly to a bellows. ATP hydrolysis then opens the complex for the next substrate [140].

Genes of ABC proteins and complexes in human, and in select mammals, are classified into seven subfamilies ABCA-ABCG. ABCE and ABCF do not act as transmembrane transporters, as they lack the TMDs. The classification was originally based solely on sequence similarity [142], but now also considers substrates and domain architecture (i.e. whether the class contains full- and/or half-transporters) [143]. A list of all human ABC transporters with their substrates and tissue distributions was compiled by Alam and Locher [143].

Another large and well-known family of primary active transporters are the rotary ATPase transport complexes. They are found across all forms of life, and are capable of pumping protons or sodium ions through biological membranes. The rotary ATPases are protein nanomachines that use a rotary motor to enable their transport function. This mechanical motion is fueled by energy from ATP hydrolysis [145]. Each rotary ATPase complex is divided into four main parts (see Figure 2.9). The first building block is a rotating barrel-like structure made from TMHs that is integrated into the membrane. This transmembrane domain contains multiple ion binding sites on its membrane-lipid-facing outside, where carbonyl groups from conserved acidic amino acids (Asp or Glu, see Figure A.1) bind to each cation [144]. Next, there is a large complex located

outside of the membrane that essentially acts as the “engine”; it converts the chemical energy from ATP hydrolysis into a rotating motion. The engine and the rotating barrel are connected by a third protein complex that represents the “drive shaft”. This peripheral structure reaches into the ATP hydrolysis domain and transfers the mechanical motion to the rotary TMH domain, to which it is connected. Finally, there is a second, smaller transmembrane domain that does not rotate, and is attached to the ATP-binding structure. This domain contains an opening that loads a cation into one of the binding pockets located on the rotary structure. The barrel then completes almost one full rotation with the cation in the binding site, until it reaches the second opening, which faces the other side of the membrane. These two *half-channels* are separated by a highly conserved (positively charged) arginine residue, which prevents the cation from remaining in the pocket [144]. Substrate specificity between proton and sodium pumps may be determined by the ratio of polar and hydrophobic residues around the interacting Asp or Glu, as well as subtle differences in structural conformation [146]. The exact protein composition and shape of the complex varies between different rotary ATPases, they are subdivided into F-, A- and V-type ATPases [144]. Some members of the F- and A-type rotary ATPase subfamilies can (or always do) “run in reverse”, meaning they use a proton or sodium gradient to turn ADP and phosphate back into ATP, in which case they are referred to as ATP *synthases*. As mentioned previously, the production of ATP is vital for powering many essential processes in the cell [144]. The A-type sub-family of ATPases/synthases, as well as those ATP synthases that rely on a sodium gradient, are found exclusively in prokaryotes [144].

In contrast to the aforementioned rotary ATPases, the P-type ATPase complexes transport a wider range of cations, either in one direction, or in both directions simultaneously. Their transport mechanism is a cycle of substrate binding in the transmembrane domain, followed by autophosphorylation after ATP hydrolysis, a subsequent conformational change leading to substrate release on the other side, and finally dephosphorylation [150][151]. These transport complexes can be made up of individual proteins [152], or of a single, continuous amino acid sequence [153]. While most of these primary active transport mechanisms are found across all forms of life, and while the essential residues and motifs are often heavily conserved across evolution, the actual substrate specificity tends to rely on substrate binding motifs or pockets, where the binding is dependent on a small number of solvent-accessible residues. For example, the P-type ATPase CtrA3 that transport copper ions across the membrane of the bacterium *Aquifex aeolius* employs metal binding domains (MDB) consisting of two cysteines, or one cysteine and one histidine [154]. On the other hand, the ion binding sites found in P-type sodium-potassium pumps contain the polar and uncharged residues Ser, Thr, Asn and Gln in the sodium binding pocket, while the potassium binding is carried out by acidic side chains in the form of Asp and Glu (see Figure 2.10) [149]. Sodium-potassium pumps follow the alternating access model, meaning that only one binding pocket is accessible at a given time, pumping sodium out of the cell, and potassium into the cell Shinoda et al. [148]. The complex either binds three sodium ions in the first conformational state E1, or two potassium ions in its second conformation E2 [149]. A charge selectivity ring of acidic residues prevents anions from entering the binding pocket [155]. Rui et al. [149] suggest that the pump has a

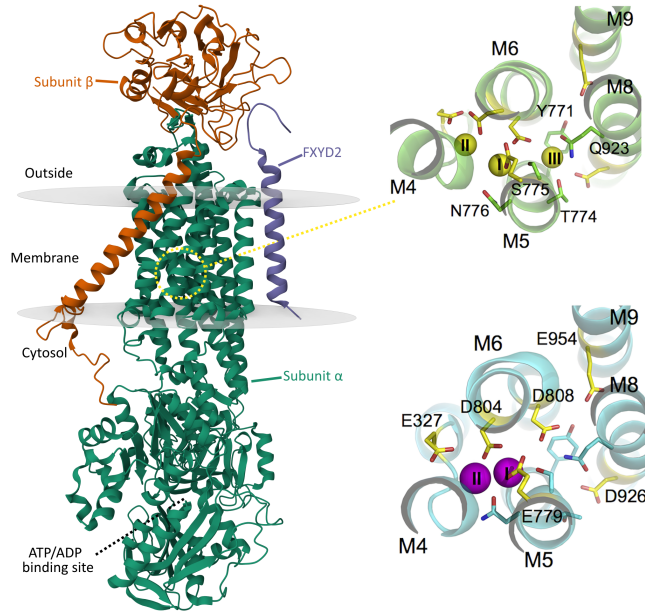


Figure 2.10: The structure of the P-type ATPase sodium-potassium pump from *Sus scrofa* (wild boar). Subunits α and β are encoded by the genes *ATP1A1* and *ATP1B1*, respectively. The protein FXYD2 acts as a regulator for the complex. The structure, which stems from Kanai et al. [147] (PDB:3WGV, created with Mol* [129]), is pictured in the E1P state, while three sodium ions are located in the binding pocket, ATP hydrolysis has taken place, and ADP is bound to the complex (not shown). The binding pocket of that complex is displayed on the top right, along with residues from interacting amino acids (see Figure A.1). On the bottom right, the same binding pocket is shown in the E2 conformation (structure by Shinoda et al. [148], PDB:2ZZE), with two potassium ions bound to the residues that are now present in the cavity, and P_i still bound to the phosphorylation site. The pictures of binding pockets were adapted from Rui et al. [149].

self-correcting mechanism, involving small structural changes, that pauses the transport if potassium ions enter the E1 sodium binding site, or sodium the E2 binding pocket, until the wrong cation is removed.

2.1.3.2 Carrier proteins

Many transmembrane transport proteins do not interact with energy carriers like ATP directly, but instead use the existing *electrochemical potential* of their substrates across the membrane to carry out their functions [156]. An electrochemical gradient is the result of an unequal distribution of electrical charges, and/or a difference in substrate concentrations, between the two environments that are separated by the membrane. This difference can be used as a source of energy for the transport of molecules, since both the charges and the concentrations are striving towards an equilibrium, due to the natural tendency of systems to minimize energy differences and maximize entropy [157]. Transporters with these types of mechanisms carry a broad range of molecules across biological membranes, including ions, sugars, amino acids and numerous drugs [158][159].

There are three types of transport systems that are driven by electrochemical

potentials. Uniporters carry one substrate in a single direction across the membrane, whereas symporters carry two molecules simultaneously, and antiporters transport their two substrates in opposing directions in an alternating fashion [156]. Symporters and antiporters use electrochemical gradients of ions to carry an additional substrate across the membrane, often against its concentration gradient. These differences in ion concentration are, in turn, often created by primary active ion transporters that pump the ions into or out of the cell, using energy from ATP hydrolysis. Examples include the rotary ATPases or sodium-potassium pumps discussed in the previous section, which use energy from ATP to create and maintain proton- and sodium/potassium gradients, respectively [157]. Uniporters, on the other hand, rely solely on conformational changes induced by substrate binding, substrate release, and sometimes lipid interactions to facilitate the transport [159][29]. This is why symporters and antiporters are classified as *secondary active transporters*, while uniporters carry out a form of *passive transport* [118][158]. Both symporters and uniporters return to their initial state after releasing their substrates on the other side of the membrane, while antiporters first need to transport their other substrate into the opposite direction [156].

The carrier-mediated transport reaction is driven by a catalytic process that is started by interactions with the substrate(s). Changes in binding energy between the molecule and residues in the binding pocket help the protein in overcoming an activation barrier, which leads to a series of conformational changes that ultimately cause the protein to open on the other side of the membrane and allow the substrate or substrates to be released [160][161]. This also means that the physicochemical properties and spatial orientations of amino acid residues inside a binding pocket are often highly specific for each transported substrate, since achieving the optimal binding energy with the transported molecule is a prerequisite for overcoming this energy barrier [162][163][160].

Interestingly, many carrier protein families actually show a relatively low sequence similarity to each other, but still form structurally conserved folds that carry out similar transport mechanisms [29]. In total, there is a small number of protein folds that a large percentage of carriers can be assigned to, the most common ones being the fold shared between members of the Major Facilitator Superfamily (MFS fold), the LeuT-like fold, plus several more folds that are less commonly observed, including those of orphan transporters where no protein with a similar fold is known [162][159][164][165]. Each fold type is made up of multiple *structural repeats*, mostly in the form of transmembrane helices connected by non-helical regions, that assemble into similarly shaped 3D structures [162]. However, the small number of interacting residues located in and around the binding pocket(s), as well as residues involved in the transport mechanism, often show conservation across proteins that transport a particular substrate, or set of substrates [162][164][161].

One of the most extensively studied MFS carrier families is that of the GLUT sugar transporters in the membranes of eukaryotic cells, which are encoded by the solute carrier gene family SLC2. They mainly transport sugars into the cell using a uniport carrier mechanism, mostly hexoses such as glucose. Up until now, 14 GLUT family members have been found throughout the human body, which differ in their tissue distribution, substrate specificity, and transport rate

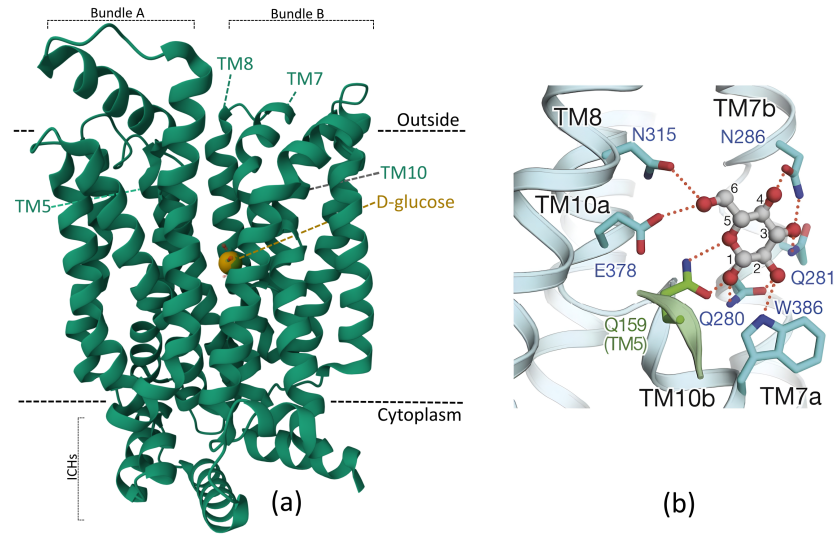


Figure 2.11: (a) 3D structure of the human GLUT3 uniporter in its outward-occluded conformation, with D-glucose located in the binding pocket. The protein structure is an example of the MFS fold, consisting of two α -helical bundles made up of six TMHs each (TMs 1-6 and 7-12) that form a binding pocket in the middle, separated by a number of intracellular helices (ICHs). Structure is from Deng et al. [166] (PDB:4ZW9), and was rendered with Mol* [129]. (b) View of the binding pocket, when looking at Bundle B from the perspective of Bundle A. The residues from various helices are positioned strategically to form hydrogen bonds with atoms in the substrate. Picture of binding pocket adapted from Deng et al. [166].

[167][168][29]. GLUT is one of three families of glucose transporters found in human cell membranes, together with the mammalian sodium-glucose symporter family SGLT, and the SWEET family of sugar exporters found in animals and plants [168][28]. As is typical for proteins that belong to the major facilitator superfamily (MFS), the GLUT transporters are made up of twelve TMHs. The helices are arranged into one N-terminal and one C-terminal helical bundle made up of six TMHs each, connected by a cytoplasmic domain that can contain intracellular helices (ICH) (see Figure 2.11a). The substrate binding pocket is located between the two bundles [169]. The GLUT family members use a transport mechanism that fits the previously discussed alternating access model, called the *rocker-switch mechanism*, where the two helical bundles rotate around the binding pocket, in order to transition between an outward-open and an inward-open conformation, with multiple steps in between [28].

The 14 proteins can be clustered into three groups, based on sequence similarity [167][168]. One of these sequence clusters is made up of GLUT1-GLUT4 [167]. While the pairwise sequence identity scores of the four transporters lie between 52.04% and 65.45% (see Figure A.2), the physicochemical properties of the amino acids, such as size, hydrophobicity and helix propensity are often conserved (see Figure A.3). This sequence *similarity*, together with similar sequence lengths of around 500 amino acids, leads to four proteins with similarly shaped 3D structures that all match the MFS fold [168][29]. The sugar binding site is a cavity located inside the membrane that is lined with highly conserved residues, stemming from multiple THMs and loops. Their positions and chemical properties allow them to precisely form non-covalent bonds with

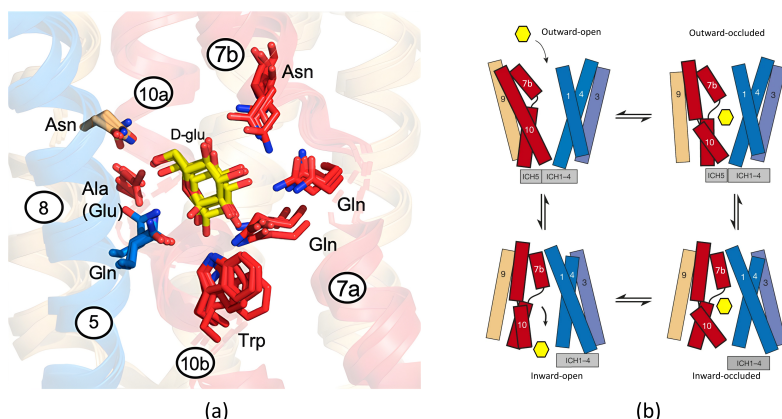


Figure 2.12: (a) Similarities between the sugar binding pockets of the human glucose transporter GLUT3 (PDB:4ZW9 [166]), the STP10 protein from the plant *A. thaliana* that also shows high substrate specificity towards glucose (PDB:6H7D [170]), the PfHT1 transporter from the malaria parasite *P. falciparum* that can carry several different sugars including D-glucose (PDB:6RW3 [171]), and the proton-xylose symporter XylE from *E. coli* (PDB:4GBZ [172]), for which glucose acts as an inhibitor. In GLUT3, an alanine from helix TM8 is replaced by a glutamic acid. While the structural conservation of the fold and binding pocket is high between the four proteins, they differ in their substrate selectivity, and also show a relatively low sequence identity to each other. Figure adapted from Drew et al. [29]. (b) The rocker-switch mechanism, as observed in the GLUT transporters and homologues. Here, the structure and mechanism of the human D-fructose transporter GLUT5 is shown, other family members and homologues show high structural similarity. Adapted from Nomura et al. [173] and Drew et al. [29].

the substrate (see Figure 2.11b), which leads to a high specificity that can exclude similarly shaped molecules, and even stereoisomers of the substrates [29].

These residues are not just conserved between GLUT1-4, but can also be found in homologues with much lower sequence identity from more distantly related organisms, including STP10 from *A. thaliana*, XylE from *E. coli*, PfHT1 from the malaria parasite *P. falciparum*, or glcP from the gram-positive *S. epidermidis*. Despite the fact that the pairwise sequence identity between GLUT1-4 and these homologues is even lower, ranging from 24.53% to 30.91% (see Figure A.2), they still form very similar folds and binding pockets, with highly conserved binding sites (see Figure 2.12a) [173][29]. Interestingly, the exact substrate specificity varies between the homologous proteins. Whereas GLUT3 and STP10 both show a very strong preference towards D-glucose, the malaria protein PfHT1 is capable of transporting a wider range of sugars that includes D-glucose, D-fructose and D-xylose. Furthermore, the D-glucose molecule even acts as an inhibitor for the *E. coli* protein XylE, which regularly transports the pentose sugar *xylose* as a proton-driven symporter [29]. It can therefore be said that the presence of this sugar binding pocket, with its conserved binding sites (see Figure A.3), can be used to classify a carrier protein as a sugar transporter that is related to the human GLUT family, while the specific set of substrate(s) that it carries might be determined by other factors. A likely candidate for this fine-tuning of the binding pocket is the transmembrane helix TM7, which is split into two parts TM7a and TM7b by a hinge-like motif. The movement of the *substrate gating helix* TM7b is important for guiding the correct molecule

into the binding pocket, and for subsequently “closing the door”, and carrying out the rocker-switch transport mechanism (see Figure 2.12b). Recent studies suggest that small evolutionary changes to TM7b composition are capable of adjusting the sugar selection for the different GLUT-related proteins [29].

Another set of highly conserved residues is related to the transport mechanism. The 3D structures of MFS transporters are typically held in their current structural conformation (i.e. outward-facing, outward-occluded, inward-occluded, inward-open, see Figure 2.12b) through salt bridges that form or break, in order to move from one conformational state to the next, which is usually initiated by interactions with the substrate(s) [28]. In GLUT transporters, there is a network of salt bridges between residues located in the ICHs and at the intracellular ends of THMs (see Figure 2.11a), which hold the protein in the outward-facing orientation [29]. These conserved residues (four arginines and four glutamic acids), together with an additional ICH that is unique to sugar carriers, have previously been used to classify MFS members as sugar transporters from their amino acid sequence before the structure was known, and are therefore called the *sugar porter motif* [174]. Binding of the substrate from the outside leads to a series of conformational changes that ultimately break those salt bridge bonds, leading to the inward-open state where the substrate can escape on the other side. Finally, the empty binding pocket then causes the protein to spontaneously fold back to the outward-facing conformation, which is helped by the energy from the re-formation of the salt bridges, and by interactions of TMHs with surrounding membrane lipids [29]. This transition has a very low energy barrier, which explains how uniporters can transport small molecules without relying on external energy sources, such as ion gradients or ATP hydrolysis [29].

Finally, some MFS transporters also contribute to multi-drug resistance (MDR) in bacteria, for example MdfA and EmrD from *E. coli* [175][31][30]. Most MFS-fold MDR pumps are driven by proton-driven antiport [161]. Similarly to the previously discussed ABC complexes that act as MDR pumps (see Section 2.1.3.1), they can also take up substrates from the membrane environment, and their 3D structures contain several binding sites in one large, hydrophobic cavity, allowing them to *selectively* export a multitude of xenobiotics, antibiotics and other substrates [30]. The structure of EmrD has been experimentally resolved, and suggests that the substrate specificity is not just determined by inward-facing residues from central TMHs, but also by two loops that reach into the cytosol-facing layer of membrane lipids, where they help with the recognition and binding of lipophilic substrates [31].

The second-most common 3D structure shared by carrier proteins is the LeuT-like fold, named after a bacterial symporter that uses a sodium gradient to carry small hydrophobic amino acids, such as leucine and alanine, across the membrane of the gram-negative *A. aeolicus* (see Figure 2.13a) [28][176]. It is part of the amino acid polyamine organo-cation (APC) superfamily of carrier proteins, which includes cation-driven symporters for various solutes, e.g. five families of human amino acid transporters, the aforementioned SGLT sugar-sodium symporters, as well as a family of anion exchangers [177][178]. APC is the second-largest carrier superfamily after MFS, and is, by its current definition, made up of 18 families [177]. The typical LeuT-like fold is made up

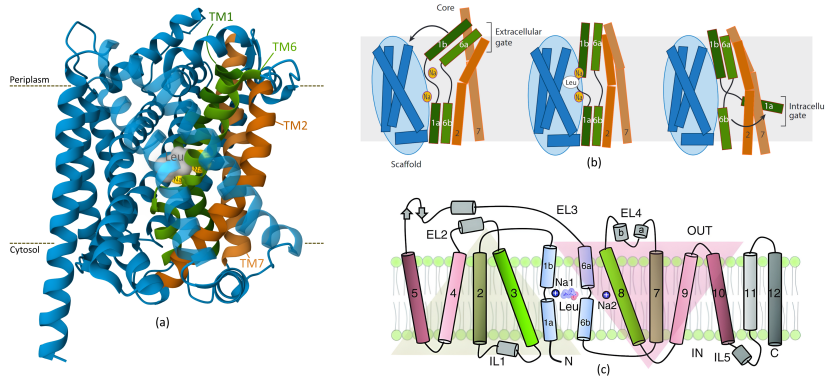


Figure 2.13: (a) Crystal structure of the symporter LeuT that carries small hydrophobic amino acids, along with two sodium ions, across the inner membrane of *A. aeolicus* (PDB:2A65 [180], rendered with Mol* [129]). The transporter is pictured in its occluded conformation, with all substrates located at their binding sites. The structure is split into two parts: the flexible core domain is colored in green and orange, while the sturdy and tightly packed scaffold domain is blue, matching the colors in the top-right panel. The two left-most helices in the picture (TM9 and TM12) form a binding interface that can connect to a second copy of LeuT, in order to create a homodimeric complex. (b) The rocking bundle mechanism, which is commonly observed among carrier proteins with LeuT-like folds. The corresponding amino acid polyamine organo-cation (APC) superfamily is the second-largest after the previously mentioned major facilitator superfamily (MFS) [177]. The presence of all three substrates in the binding pocket allows the extracellular gate to close, leading to conformational changes in the flexible core domain that ultimately releases the substrates into the cytosol [28]. The colors of helices and substrates match those in the left panel. Figure adapted from Drew and Boudker [28]. (c) The 5+5+2 LeuT-like fold, as found in the amino acid symporter LeuT from *A. aeolicus*, where it was first discovered [181]. The structural repeats, which form the binding pocket in their middle from partially unwound helices, are shown as triangles. Adapted from Penmatsa and Gouaux [181].

of two structurally inverted repeats containing five TMHs each, plus two additional C-terminal TMHs that help in holding the protein together (see Figure 2.13c) [164][28]. This fold is referred to as the 5+5+2 TMH topology [177]. Other TMH arrangements, such as 5+5+4 or 7+7, exist as well among APC superfamily members [177], and were shown to be homologous to the 5+5+2 LeuT-like fold [179].

The LeuT protein itself is divided into two structural repeats, encompassing TMHs 1-5 and 6-10, respectively. Helices 1 and 6 are connected to each other in the center of the structure, where they are partially unwound in the middle, forming a flexible binding pocket for its three substrates (two sodium ions and one L-leucine). The two TMHs 9 and 12 form a binding interface that connects LeuT to a copy of itself, as these proteins typically occur as homodimers (see Figure 2.13a,c) [180]. Interestingly, while the two structural repeats are so similar to each other that they can be superposed on top of each other with a root mean square deviation (RMSD) of only 5.3 Å, their sequence similarity is so low that the symmetry was only detected when the 3D structure was solved [180]. This is also the case for LeuT-fold proteins in general, which, similarly to the MFS proteins, are highly divergent in their amino acid sequences, while showing conservation in places related to the mechanism or substrate binding [182]. Residues in cation binding sites of secondary active transporters with

a LeuT-like fold are conserved [28], and structural superposition of multiple APC sodium symporters showed similarly shaped substrate binding pockets [183][184][185]. Chemical differences between substrate molecules of two homologous LeuT transporters are reflected by evolutionary changes in binding pocket amino acid residues [181][182]. LeuT/APC proteins employ a transport mechanism similar to the rocker-switch model used by MFS carriers, called the *rocking-bundle mechanism*. A *core domain*, consisting of flexible, partially unwound TMHs, is responsible for transitioning between inward- and outward-facing states, while the remaining TMHs form a *scaffold domain* that provides rigidity to the structure (Figure 2.13a) [28].

2.1.3.3 Channels and Pores: Overview

The third major category of transmembrane transport proteins encompasses channels and pores [118]. Most protein families of this type utilize *energy-independent* passive transport to carry their substrates [57].

In this section, we will use the term *channel* for passive α -helical bundle transporters of ions (ion channels), and for certain families of water channels that are closely related to ion channels in terms of structure and mechanism (aquaporins). On the other hand, proteins that are referred to as *pores* or *porins* are outer-membrane pore (OMP) transporters that possess the β -barrel fold, a type of super-secondary structure made up of β -sheets that is found almost exclusively among OMPs (Figure 2.5c). This distinction is in line with the TCDB classification system for membrane proteins (1.A.X for channels, 1.B.X for pores) [118]. Porins are found in the outer membranes of gram-negative bacteria, as well as in the outer membranes of eukaryotic cellular components that have an evolutionary relation to ancient bacterial species, such as mitochondria and chloroplasts [109][186]. Pores can be more or less selective towards their substrate, and are, overall, capable of carrying a much wider range of molecules than the α -helical ion channels [57].

Whereas α -helical bundles can form many different folds, such as the MFS and LeuT folds discussed in Section 2.1.3.2 or the four domains found in every ABC transporter (Section 2.1.3.1), all β -barrels share a similar cylindrical architecture, where the average diameter of the pore is determined by the number of included β -strands [57]. The sequence regions between the β -strands can form domains that reach into the pore, and help with substrate selection [187]. As observed in transmembrane structures made up of helices, the outward-facing amino acid residues found in pores are more likely to be hydrophobic, while the inward-facing side chains are more hydrophilic, which allows water molecules to line the inside, while the outside of the structure can be integrated into the hydrophobic membrane environment [57]. The barrels are often made up of a single peptide chain that forms alternating, anti-parallel β -strands (Figure 2.3) [109][188].

Channels and pores both possess a variety of different gating mechanisms that can open or close the protein for substrates, or influence the flow rate in some way. This includes voltage-gated channels that are regulated by the membrane potential, ligand-gated channel proteins that open or close based on the presence and binding of a specific molecule to the protein, and many more, including

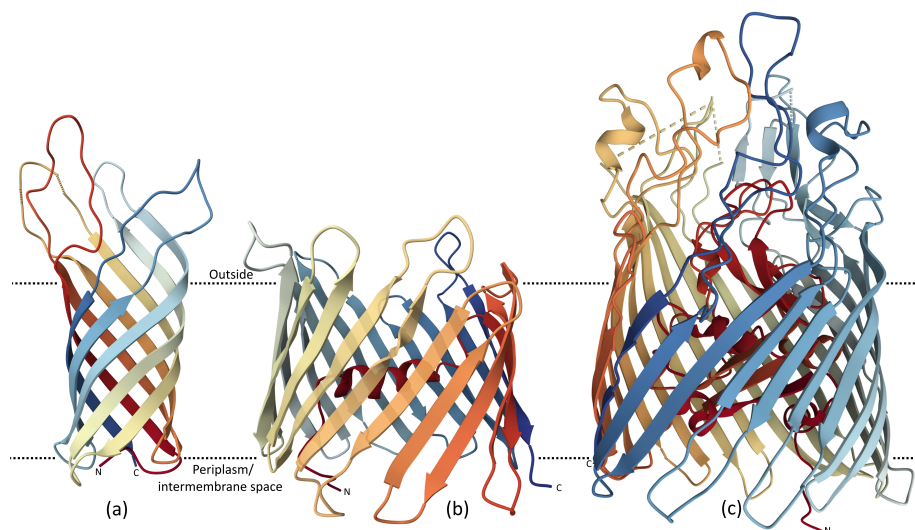


Figure 2.14: Three β -barrel porins, of varying sizes. The structures are colored by sequence position, ranging from red on the N-terminal start, to beige and then blue. All structures were rendered with Mol* [129]. (a) β -barrel domain of the OmpA protein, found in the outer membrane of *E. coli* (PDB: 1BxW [191]). (b) The human voltage-dependent anion channel VDAC1 in the outer mitochondrial membrane (PDB: 2JK4 [192]). In contrast to the other two proteins, the outside represents the cytoplasm, and the other side is the intermembrane space (IMS) of the cellular component. (c) Crystal structure of the ligand-gated siderophore transporter FepA from the outer membrane of *E. coli* (PDB: 1FEP [193]).

lipid gating, light gating, temperature gating, and mechano-sensitive gating [57][189][190].

2.1.3.4 β -Barrel Porins

The transmembrane domain of OmpA (short for *Outer membrane protein A*) in *E. coli* is an example of a small β -barrel pore (Figure 2.14a). It is only made up of eight β -strands, and is gated by a single salt bridge that crosses its center [191]. Binding the two interacting residues (Glu52 and Arg138) to alternative interaction partners in their vicinity (Lys82 and Glu128) opens the pore [194], which allows both ions and small, water-soluble molecules to enter or exit the space between the inner and outer membrane of *E. coli* at a relatively slow rate [195][196]. On the other hand, it is also the third-most abundant protein in the entire proteome, when measuring the protein synthesis rates [197]. The substrate specificity of the pore might be determined by the polar side chains that line the inside, by its quaternary structure, and by its diameter [194]. The periplasmic domain of OmpA (not pictured) is assumed to bind non-covalently to the peptidoglycan layer [198].

The Ferrienterobactin receptor (FepA) protein from *E. coli*, on the other hand, is an example of a large pore, as its β -barrel structure is made up of 22 β -strands (see Figure 2.14c) [193]. FepA is responsible for importing siderophores. These small organic molecules are released into the extracellular environment by *E. coli*, where they selectively bind to iron ions, and are subsequently imported back into the cell to release their bound iron and maintain iron homeostasis

[199]. In contrast to the small OmpA pore, this protein, in addition to its pore-forming transmembrane domain, also contains an N-terminal globular domain, which is located inside of the opening created by the large pore [193]. FepA is ligand-gated; the transport requires the presence of a small protein called TonB, as well as conformational changes caused by the substrate itself [200][201]. Extracellular, elongated loops between the pore-forming β -strands also play a role in the transport mechanism. Two Arginines at sequence positions Arg286 and Arg316, located in the fifth extracellular loop of the transmembrane domain, have been found to contribute to substrate specificity, by binding to the transported substrate [199][202]. Recent studies suggest that after binding of TonB and a siderophore molecule, a conformational change takes place that causes the central globular domain to exit the pore into the periplasm, where it subsequently releases the substrate (“ball-and-chain mechanism”) [200][203]. This makes the mechanism of FepA relatively unusual for outer membrane porins, and its dependence on TonB could classify it as a type of *energy-dependent* transporter, although the specifics around how energy is transferred to the transport reaction remains unknown [201][203].

The voltage-dependent anion channel (VDAC1) in human mitochondria is an example of a β -barrel pore that occurs in eukaryotic cells. It is made up of one relatively short, N-terminal α -helix that sits in the pore, followed by 19 β -strands that form the barrel (see Figure 2.14b) [192]. The pore changes from its open state to a closed state when the membrane potential deviates from 0 mV, where it is maximally open, by more than 30 mV [204]. While the pore is capable of transporting a wide range of small hydrophobic molecules and ions, it also shows a selectivity towards some substrates, and is capable of excluding other molecules completely, even if their size and hydrophobicity match its known ligands. ATP shows a higher flux compared to molecules of similar size and charge, which makes sense considering the localization in the mitochondrial outer membrane. This selectivity is likely caused by charged residues inside the pore that the substrates interact with when the structure is in its open conformation [205][206].

An interesting example for how substrate selection happens in transmembrane pores is Maltoporin (lamB), also from the outer membrane of *E. coli*. This β -barrel pore forms a homotrimer of three pores, with 18 β -strands in each individual protein (Figure 2.15) [207]. Despite its relatively large diameter, the pore still manages to be selective towards long malto-oligosaccharides, which do not travel through non-specific pores [187][207]. There are two main reasons for this: On one hand, the space inside the channel is constricted by three loops that fold into the pore, narrowing the path that molecules can take [208]. On the other hand, the inside of the pore is lined with several hydrophobic, aromatic residues (Trp74 from the neighboring pore, Tyr41, Tyr6 and Trp420) that form a so-called *greasy slide*, which enables apolar van der Waals interactions with the substrate in order to pull the molecule into the pore and guide it through to the other side (Figure 2.15a,b) [187]. The inside is also lined with appropriately positioned side-chains that can form hydrogen bonds with the oligosaccharide (i.e. Arg8, Arg33, Arg82, Arg109, Asp116, His113 and Glu43) (Figure 2.15a) [187].

In conclusion, we can say that β -barrel pores, which can be found in the outer



Figure 2.15: The *E. coli* outer membrane pore maltoporin, as seen from the outside of the membrane (PDB: 1MPO [187], rendered with Mol* [129]) (a) An individual maltoporin protein with a maltopentose oligosaccharide located in its center. Residues that can form hydrogen bonds with the substrate are marked in blue, the side chains that engage in van der Waals interactions are colored yellow. The Trp74 side-chain on the top-right reaches into the pore of the neighboring maltoporin in the complex. (b) The homotrimeric complex that maltoporin forms in the membrane. Maltopentose is bound to each pore in the complex, the “greasy slide” residues that interact with the substrate are marked in yellow.

membranes of gram-negative bacteria and related eukaryotic organelles such as mitochondria and chloroplasts [188], are a very diverse group of membrane proteins. They can be small or large in diameter [191][193], be very specific or broad in their substrate selection [195][207], and can be regulated by different gating mechanisms that can depend on ligands, lipids, the membrane potential, temperature, shear stress, among others [109][57]. A broad, initial substrate selection takes place through the diameter and shape of the barrel [194], and can then get narrowed down by loops [202] or other domains [200] that are located in and around the pore, in order to limit the three-dimensional structure and chemical composition of the transported molecule. Substrate specificity is typically caused by the positioning and the physicochemical properties of a small number of residues [187][206][199].

2.1.3.5 Ion Channels and Aquaporins

Ion channels are energy-independent and passive transmembrane transport proteins, where the transmembrane domain is mainly made up of an α -helical bundle. The vast majority of channel proteins carry monoatomic ions, typically with a high substrate specificity. Most channels can be opened or closed through different gating mechanisms (see Section 2.1.3.3) [209][57]. The flux rates of substrates through channel proteins is often very high compared to other forms of transport, with differences on the scale of several orders of magnitude [33][57]. Channels typically achieve their substrate specificity through a *selectivity filter* (SF), which is made up of a small number of residues that line a narrow part of the channel, where they can interact with the transported ion, and with surrounding water molecules [33]. Different selectivity filters allow the passage of specific organic ions, such as potassium [210], sodium [36], chloride [211], protons [212] or calcium [213]. Nonselective ion channels, e.g. the

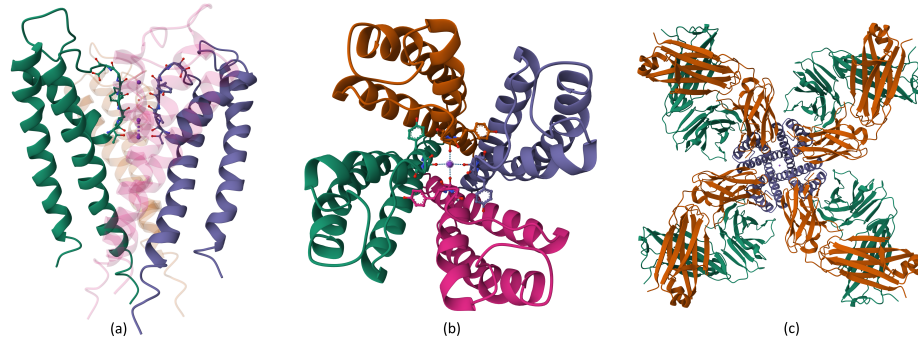


Figure 2.16: The crystal structure of the potassium channel KcsA from *S. lividans*. Figures were rendered with Mol* [129]. (a) The channel tetramer of the complex, as it is integrated into the membrane, with the cytoplasmic ends of the proteins facing down, and the extracellular loops directed upwards. Two of the four proteins are made transparent, to better see the selectivity filter, which is populated by three potassium ions (purple dots) that are separated by a water molecule (red dot) (PDB: 1BL8 [215]). (b) The same tetramer when looking down at the membrane from the extracellular side. The ions and water molecules are held in place by backbone carbonyl groups that point into the middle of the protein. Residues from the filter, and from a short helix behind it, interact with the neighboring proteins to hold the complex together (Tyr78 is shown). (PDB: 1BL8 [215]). (c) The full KcsA-Fab potassium channel complex, including the extracellular domains (PDB: 1K4C [35]).

sodium-potassium channel NaK that conducts both sodium and potassium, can achieve their lower selectivity through structural flexibility in and around their SF [214].

One example of such a filter can be found in voltage-gated potassium channel KcsA from the gram-positive bacterium *S. lividans* (see Figure 2.16). Its mechanism has been studied extensively, as it was the first channel of its kind that had its 3D structure determined through x-ray crystallography [215]. KcsA is the pore-forming protein in the KcsA-Fab potassium channel complex, where the transmembrane channel domain is made up of four symmetrically arranged copies of the KcsA protein (see Figure 2.16b). Further proteins, which are mostly made up of β -strands, coils and short 3-10 helices, are attached to the complex on the outside of the membrane (Figure 2.16c) [35].

The substrate selection is carried out by a loop between two TMHs of the KcsA transmembrane channel protein (Figure 2.16a). Each of the four proteins in the tetrameric channel complex contributes one of these loops, and together they form a narrow opening that is lined with backbone atoms from six amino acids (Thr75-Val76-Gly77-Tyr78-Gly79-Asp80) (Figure 2.16b) [33]. Both the tetrameric structure of the voltage-gated potassium channels and the sequence of their selection filter have been conserved throughout evolution [210]. A similar potassium selectivity filter motif can be found in channel proteins from many different organisms, for example in Kch from *E. coli* [216], and in several eukaryotic potassium channels, e.g. KCNA1 (prev. $K_V1.1$) [217] and KCNA2 (prev. $K_V1.2$) [218]. In many cells, potassium channels are responsible for regulating the membrane potential, and are crucial for the function of muscles and neurons in animals [217].

Voltage-gated potassium channel proteins have a high selectivity for their sub-

strate. Even sodium ions, which are relatively similar in size and charge, are selected against with an accuracy of more than 99.9% [210][219]. This is especially important in neurons and muscle cells, where a temporary selective permeability that allows potassium, but not sodium to pass through the membrane is crucial for their function [217][57].

When ions are dissolved in watery solution, they interact with their surrounding water molecules, i.e. they are *hydrated*. The narrowness of the path through the selectivity filter requires the ion to be mostly dehydrated before it can enter [35]. The backbone carbonyl groups of the amino acids that make up the selectivity filter provide four evenly spaced oxygen atoms, thereby creating four non-covalent binding sites for (mostly) dehydrated potassium ions [220]. Two or three substrate molecules at a time can pass through the filter in a single file, with ions being separated by water molecules that enter the channel (see Figure 2.16a)[221]. Interestingly, this means that the chemical properties of amino acid side chains do not play a direct role in the substrate selection of this type of channel. Instead they have a structural function, holding the tetramer together in such a way that precisely positions the backbone carbonyls into the pore where they can interact with water molecules and potassium ions, with the help of a short helix behind the filter. For example, Tyr78 that is shown in Figure 2.16b forms hydrogen bonds with Trp68 and Thr72 from the short helix of the next protein in clockwise direction [215].

One explanation for why KscA is highly selective towards potassium, while simultaneously excluding relatively similar sodium ions, is referred to as the *snug fit* mechanism. The idea is that the distances between oxygen atoms in the selectivity filter resemble the oxygen atoms in the water molecules that make up the hydration shell of a potassium ion. These four oxygens are spaced too far apart for the smaller sodium ion, making the dehydration and subsequent non-covalent binding of potassium ions energetically favorable in comparison, as the free energy cost of partially dehydrating the sodium ion is higher than the free energy gained by its interactions with residues in the binding pocket [222][219]. More recent studies, with help from molecular dynamics simulations, have argued that the real mechanism is likely more complex than that, and might also involve the dynamic movement and flexibility of the folded amino acid sequence [219][223][33]. The hydration levels of cavity residues around the selectivity filter could play a role in regulating the channel's activity, with dehydration acting as a gating mechanism [224].

Another interesting family of channel proteins is that of aquaporins. They are closely related to ion channels, but instead have the task of transporting water molecules through cell membranes, at high flux rates. Some channels of this type also carry other substrates, such as O_2 , CO_2 , H_2O_2 , or small molecules such as glycerol [226]. The human AQP family includes 13 members that differ in their tissue distribution and secondary substrates, and show small structural variations [226]. While sequence similarity is low between members of this protein family, they all share a similar three-dimensional fold made up of six transmembrane helices. Four identical AQP proteins occur as a tetramer in the membrane that can form five channels: one through each individual protein, and a fifth one through the center of the tetrameric complex (see Figure 2.17a) [227][228][226].

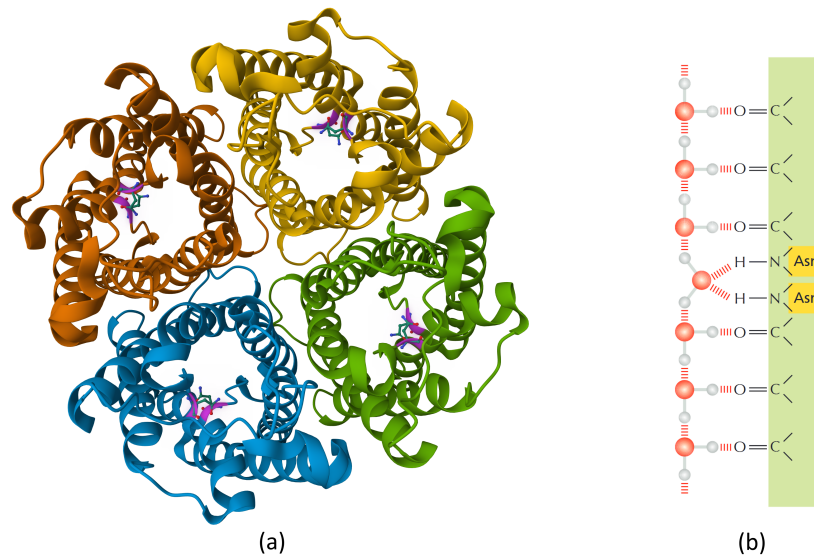


Figure 2.17: (a) Tetrameric complex created by four copies of the human AQP4 protein (PDB: 3GD8 [225], rendered with Mol* [129]). The perspective is from the outside of the cell, looking into the channels. Two highly conserved asparagines from the NPA motifs that help in orienting the transported water molecules to prevent proton leakage are marked in purple, and their side chains are shown. Directly above the upper asparagine that is closer to the viewpoint, located on the same helix, is the arginine from the ar/R pore constriction, which also includes multiple aromatic residues from helices on the opposite side (both not shown). Together, these side chains form a ring that restricts the width of the channel. (b) The water binding sites found in aquaporin channels. The two central asparagines help in orienting the water molecules in the channel, together with water binding sites formed by backbone carbonyls and hydrophobic side chains that line the rest of the pore. This prevents hydrated protons from traveling through with the chain of water molecules, by forcing them to only bind to each other or the binding sites. Figure b adapted from Alberts et al. [57].

Substrate selectivity in the four channels formed by the individual proteins is achieved by pore-lining residues in two different places. First, there is the *ar/R pore constriction*, which is made up of one highly conserved arginine (one-letter-code R) that is surrounded by several aromatic (ar) residues. This constriction, located in the region of the channel closer to the extracellular environment, is mainly responsible for limiting the width and shape of the pore to only fit its substrate(s), preventing hydrated ions and larger solutes from entering [227]. Variations in substrate selection between different aquaporins that allow additional substrates to enter the channel often coincide with changes to the *ar/R pore constriction* [227][229][226]. The second filter, located directly behind the ar/R constriction (see Figure 2.17a), prevents proton leakage, which would occur when a proton forms hydrogen bonds with two surrounding water molecules, and then follows them through the channel. Proton gradients across the membrane are important energy sources for many cellular pathways (see Sections 2.1.3.1, 2.1.3.2), meaning that protons need to be prevented from being passively transported by aquaporins. This is achieved by an intracellular and an extracellular loop that both fold into the cavity, where they meet in the middle. On each loop there is a sequence motif made up of one highly conserved asparagine, typically followed by a proline and an alanine. The motif is named

after the one-letter codes of its amino acids (NPA) [230]. The two asparagines are located in the middle of a chain of backbone carbonyls that form multiple water binding sites, while the rest of the cavity is covered by hydrophobic side chains [225]. Together, these residues in the channel manage to dictate and restrict the orientation of the water molecules as they traverse the channel, in single file. The central water is forced into its orientation through the two hydrogen bond donors in the asparagine, while the surrounding water molecules then form hydrogen bonds with the backbone carbonyl oxygen atoms or other water molecules, meaning that none of them have any valence electrons available that could bind to a proton, and carry it through the channel (see Figure 2.17b) [231].

The central pores of AQPs are speculated to carry ions, gases and other substrates, but further research is needed [232][226].

2.1.3.6 Other transport mechanisms

Other categories of transporters that are mentioned by the Transporter Classification Database (TCDB) include group translocators, electron carriers, and membrane transporter metabolons (MTM) [233].

Group translocation (TCDB 4.X) is a form of transmembrane transport where a chemical reaction involving the substrate is energetically coupled to the transport reaction. This includes, for example, the bacterial phosphotransferase system (PTS) that couples sugar phosphorylation to the uptake of sugar across the cell membrane. The phosphate group stems from phosphoenolpyruvate (PEP), which turns into pyruvate after the reaction. PEP-PTS complexes can be divided into three different parts. The first part is a transmembrane domain, made up of one or two α -helical proteins that form a binding pocket, similarly to the carriers discussed in Section 2.1.3.2. Connected to these integral membrane proteins are two intracellular domains: The *phosphotransferases* remove phosphate groups from surrounding PEP molecules, in order to pass them on to *energy coupling proteins*, which phosphorylate the substrate while it is located in the binding pocket [234][235]. The transport mechanism is similar to the previously discussed *alternating access model* (see Section 2.1.3.2), where the protein transitions between outward- and inward-facing conformations to facilitate the binding and release the transported molecule. The key difference is that the substrate undergoes a chemical reaction while it is located in the binding pocket, changing the chemical characteristics of the substrate and possibly initiating further structural changes [235][28]. Four different PTS superfamilies exist for the transport of glucose/fructose/lactose, dihydroxyacetone (DHA), galactitol/ascorbate and mannose, respectively [234].

Transmembrane electron carriers (TCDB 5.X) transfer either one or two electrons (e^-) from an electron donor molecule located on one side of the membrane to an electron acceptor on the other side. In other words, they essentially are oxidoreductases (i.e. enzymes that catalyze redox reactions) that are made up from TMHs, and located in the membrane [233]. One example is the eukaryotic cytochrome (Cyt) b_{561} family, found in plants and animals. They oxidize (i.e. take electrons from) L-ascorbate (Vitamin C), and then transfer the electron to another substrate on the other side of the membrane. In terms of protein

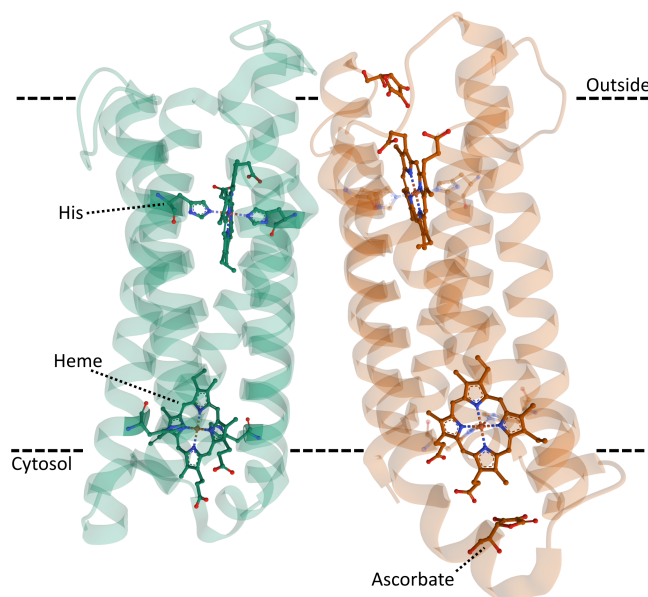


Figure 2.18: Cytochrome b_{561} transmembrane electron transfer complex from *A. thaliana*. Both proteins contain two heme groups, each of which is held in place by two fully conserved histidines that form non-covalent bonds to the central iron atom. The histidines are highlighted in the left protein, while the right protein interacts with two ascorbates, which the complex uses as electron donors. The electron is then passed on to the other side, where it can reduce another molecule (PDB: 4O7G [101])

structure they are members of the major facilitator superfamily (MFS) that was previously mentioned in Section 2.1.3.2 [236]. In *A. thaliana*, two very similar proteins with nearly identical structures called Cyt b_{561} -A and Cyt b_{561} -B can either occur as homodimers, or form a complex with each other, resulting in the characteristic twelve TMHs of an MFS transporter [101]. However, the mechanism is different from the rocker-switch model commonly found in solute-carrying MFS members [29]. Each of the two proteins in the complex contains two binding pockets, one on either side of the membrane. Every individual binding pocket features a heme group, which is held in place by two highly conserved [237] histidines, and surrounded by positively charged residues that likely bind and guide the ascorbate (see Figure 2.18) [101]. After substrate binding on the cytoplasmic side, an electron (along with a proton) is removed from the ascorbate molecule, and passed on to the intracellular heme group. Next, the electron is transferred to the extracellular heme group through a series of covalent and non-covalent bonds, likely involving a conserved phenylalanine residue that is located inside the protein, and possibly a water molecule that is bound to other residues in the center. Finally, a now-reduced atom in the outer heme group can pass the electron on to an extracellular molecule, either to a triply charged iron ($Fe^{3+} \rightarrow Fe^{2+}$), or to a monodehydroascorbate radical that reacts with the electron, and an additional proton, back to ascorbate [101]. The four histidines are conserved across 107 members of the b_{561} family. A small number of additional residues show conservation mostly in their respective sub-family [237]. Tsubaki et al. [237] classified the family into seven subfamilies: three in animals, two in plants, and one each in insects and fungi. Su and Asard [238]

found that out of 20 conserved or partially conserved residues in mammalian b_{561} proteins, six were essential for their function (including the four histidines), and another six affected the activity in some way.

Membrane transporter metabolons (MTM, TCDB 6.X) are temporary protein complexes made up of membrane proteins that transport a substrate, and enzymes that catalyze chemical reactions with the transported molecule [233][239]. The metabolon made up of GLUT4 (see Section 2.1.3.2) and either hexokinase II or glucokinase is an example of an MTM that phosphorylates glucose during import into the cell [240]. This increases the efficiency of multiple glucose metabolism pathways, which typically start with phosphorylation of the sugar molecule, by preventing back-flow of glucose through closely located exporters, and by carrying out the first step in its metabolism directly during the import [239]. Other examples of transport proteins that are part of MTMs are VDAC β -barrel pores in the mitochondria of *A. thaliana* [241] (see Section 2.1.3.4), and the human Anion Exchanger I antiporter [239][242] (see Section 2.1.3.2).

2.1.3.7 Summary and Conclusions

In summary, we can say that membrane transporters often contain a small number of highly conserved sequence positions that are related to the transport mechanism or the substrate specificity, while the remainder of the sequence typically shows relatively low sequence identity, even between closely related proteins with similar three-dimensional folds and substrate(s). However, when trying to encode these proteins as numeric vectors, to then assemble these fingerprints into datasets that can be used for the training of predictive machine learning models, there are some notable exceptions that have to be considered.

We found that many transporters work similarly to enzymes, meaning they contain binding pockets where the side chains match the substrate (e.g. Figure 2.11b, 2.13a,c). Binding of compatible molecule(s) in the binding sites then catalyzes a transport reaction, typically involving a series of conformational changes in the protein structure that open the protein on one side and close it on the other, which is referred to as the *alternating access model* (Figure 2.12b, 2.13b). This includes carrier proteins, as well as many primary active transporters, such as P-type ATPases and several types of ABC complexes (see Sections 2.1.3.2 and 2.1.3.1, respectively).

When looking at the GLUT sugar transporter in detail in Section 2.1.3.2, we found that they all share the same fold, mechanism, and binding pocket (Figures 2.11, 2.12). However, not all members of this family carry the same substrates. Some are, for example, specific towards either fructose or glucose, while others can carry both, and further homologues are compatible with additional hexoses and pentoses. While the presence of the conserved sugar binding pocket can classify a protein as a sugar transporter, the fine-tuning for one specific hexose or another depends on smaller details, such as the exact positioning of one particular residue on a flexible gating helix that can move while the protein is in its outward-facing structural conformation. This means that it could be more challenging to find a protein encoding that also includes these minute differences, in addition to problems that arise from the lower number of available training

samples for e.g. D-fructose carriers, compared to a dataset of all hexose- or all sugar transporters.

Further exploration of carrier protein 3D structures showed us that a large number of these transporters assemble into either the MFS fold, or the LeuT-like fold found in proteins from the APC superfamily. While the MFS transporters show higher structural conservation, the APC proteins are more diverse in their TMH topology. Most APC members are cation-driven secondary active transporters, with highly conserved cation binding sites that are located in a flexible part of the protein, while the remainder builds a stiff scaffold domain (see Figure 2.13a,b).

Interestingly, we found that even multi-drug transporters with a large number of known substrates, both MFS members and type IV-V ABC complexes (see Sections 2.1.3.2 and 2.1.3.1, respectively), can be highly selective towards their respective set of substrates, excluding similarly shaped molecules, even stereoisomers. This is achieved through large binding pockets with binding sites for multiple substrates (e.g. ABCB1 in Figure 2.7), which can also overlap. Many of these multi-drug transporters can open their binding pocket towards the membrane environment, where they take up their lipophilic substrates. Transporters of drugs and xenobiotics are often hydrophobic on the inside, while solute and ion transporters commonly possess aqueous pores lined with or surrounded by water molecules; this could be useful information for classification models.

The rotary ATPases from Section 2.1.3.1 have a unique type of protein structure, with conserved proton/sodium binding sites on the outside of their rotating barrel, two half-channels located in the stationary transmembrane domain, and a large complex outside the membrane that converts energy from ATP into mechanical motion (Figure 2.9). In most organisms, including animals, these pumps exclusively transport protons, while some bacterial species also contain sodium-transporting variants. In animal or plant models, these proteins could generally be classified as proton pumps, as there is no evidence of rotary sodium pumps in these organisms, and their structures are easily recognizable.

The substrate binding pocket is not the only place in the protein where amino acid conservation can occur; the conserved residues can also have important mechanistic or structural functions. GLUT proteins, as discussed in Section 2.1.3.2, contain multiple highly conserved arginines and glutamic acid residues that form a network of salt bridges to keep the protein in its outward-facing conformation. Subsequent breaking of these bonds then changes the protein structure towards its inward-facing state (Figure 2.12b). Proteins from this transporter family also contain the *sugar porter motif*, a short part of the sequence that can be used to distinguish their sequences from other members of the major facilitator superfamily. This motif is, however, not related to the actual binding pocket, but rather has structural functions related to the mechanism, and is located on an intracellular domain. Analysis of several GLUT proteins and homologues also led to another interesting observation: While some residues are conserved across the entire tree of life, others are found only in certain types of organisms, such as plants, fungi, animals or bacteria. For example, some additional binding sites that were conserved in eukaryotic GLUT transporters could not be found in bacterial homologues (Figure A.3), meaning that residue

conservation can also just occur between proteins whose organisms stem from the same family, class or domain.

Another example of mechanism-related residues were the eight perfectly conserved histidines found in Cyt b_{561} electron transport complexes (Section 2.1.3.6, Figure 2.18), which are responsible for holding four heme groups in place. The fact that each protein in the homodimer has four highly conserved histidines in a protein sequence of around 250 amino acids, along with residues associated with hydrophobic transmembrane helices, could be useful information for statistical models that aim to find similar proteins in a proteome.

We also discovered that in some cases, the binding pocket of a transporter does not directly match the substrate that it is annotated with, but instead interacts with a carrier molecule that delivers the substrate to the protein. This includes the *E. coli* iron siderophores that get released into the extracellular environment, where they bind to elemental iron, and are subsequently imported back into the periplasm by the iron importer FepA (see Section 2.1.3.4, Figure 2.14c). Other examples of carrier molecules include the electron donors and acceptors that interact with transmembrane electron transporters described in Section 2.1.3.6.

Transporters often occur as large complexes made up of several proteins, where each member of the complex has a particular task, centering around the transmembrane transport domain, which is made up of bundled α -helices, or a β -barrel structure. Primary active transporters tend to contain ATP binding domains that transfer external energy from ATP hydrolysis to the transport mechanism, like the two nucleotide binding domains (NBDs) that are present in each ABC transporter (Section 2.1.3.1, Figures 2.10, 2.7 and 2.6). Other transport complexes contain enzymes, for example the energy coupling proteins in the PEP-PTS group translocation system that phosphorylates sugars while they are located in the binding pocket of the carrier domain (Section 2.1.3.6). Additional proteins can have structural functions and connect different functional domains, or act as regulators that activate the transport, such as the FXYD2 protein in Figure 2.10.

In Section 2.1.3.1, we also discussed some complexes that are modular, meaning that they can change their substrate specificity by temporarily or permanently recruiting further proteins to the complex. Instead of the regular two transmembrane domains (TMD) that are structurally similar to each other, type III ABC complexes can contain one of several *S-components* that each bind a different substrate, along with a second TMD that is structurally dissimilar. In type II complexes from the same family, the substrate selection is done by a substrate binding protein (SBP) that temporarily docks to the transporter to deliver the substrate into the internal cavity, which is lined with a slippery surface instead of containing a binding pocket. However, binding pockets can be found in complexes with the ABC mechanism, for example they are common in type I complexes (like MalFGK₂, see Figure 2.6) where the SBP is, in turn, less specific towards its substrate. The MacAB-TolC tripartite multidrug efflux pump (Figure 2.8) was another interesting example, as it contained an inner-membrane domain that transfers mechanical motion to a periplasmic domain, which then takes up its substrate and pumps it through an outer-membrane pore that is also part of the complex, meaning that ATP hydrolysis, substrate uptake and transmembrane transport are carried out by three different complexes that

bind together. Another observation when comparing transport complexes from different organisms to each other was that they often go through gene fusion events during evolution, meaning that an ABC transporter that is made up of four different proteins in *E. coli* can be encoded by only one or two sequences in human, for example. This might become important when adding proteins from both organisms to the same dataset.

Ion channels and aquaporins, which we explored in Section 2.1.3.5, employ a different type of transport mechanism, where the channel is constricted by selectivity filters (SF) that are encoded by highly conserved sequence motifs. These SFs form narrow passages involving several binding sites, made from backbone carbonyl groups of amino acids in the sequence(s) (Figure 2.16a), and sometimes interspersed by individual conserved side chains that interact with the passing molecules, e.g. the NPA motif in aquaporins (Figure 2.17a,b), or the Asp-Arg SF found in voltage-gated proton channels [212]. Interestingly, many of these motifs do not interact with the substrate through their side chains, but instead cause the complex to fold in a particular way that precisely positions the backbone carbonyl groups into the pore to form binding sites, and/or constrict the opening of the channel to only allow molecules of a particular size and shape to enter (e.g. the ar/R constriction in aquaporins). According to the *snug fit* theory, the actual selection is carried out by the distances between oxygen atoms in the selectivity filter, which precisely re-create the hydration shell of the transported ion (Figure 2.16b), thereby making its dehydration and subsequent transport energetically favorable, compared to other hydrated ions. Furthermore, studies involving molecular dynamics simulations suggest that the flexibility of SFs could also play a role in their substrate specificity, which would be harder to encode from sequences or crystal structures. Regardless of the actual mechanism, ion channels and aquaporins do contain conserved residues that correlate with their substrate, which could be enough information to predict the transported ion from an encoded protein sequence.

When looking at β -barrel pores in Section 2.1.3.4, we found that some porins are non-selective (Figure 2.14a), while others can interact with and select for their substrates through pore-lining residues, and through globular domains or loops that fold into the pore. Substrate-specific porins can have some unique mechanisms, like the greasy slide in maltoporin that pulls long oligosaccharide chains through the outer membrane while still achieving substrate selectivity with a hydrogen bond network (Figure 2.15a), or like the ball-and-chain mechanism of FepA that releases a large globular structure into the intracellular environment after substrate binding (Figure 2.14c). The VDAC1 transporter (Figure 2.14b) highlights the structural and functional similarities between bacterial and human mitochondrial porins. While residues in selective β -barrel pores do interact with their substrates, and might therefore add information about their substrates to the protein sequence, the fundamental structural and mechanical differences between porins and α -helical membrane transporters could pose a challenge when trying to find similarities between proteins carrying the same substrate, as the different frequencies of amino acids in the two types of secondary structure [58] could become a confounding variable in the dataset.

The goal of this thesis is to mathematically encode transport protein sequences, and then use those encodings to train predictive models that can categorize the

transporters according to their substrates. Good starting points for these encodings are the binding pockets found in many proteins, as the conserved side chains they contain tend to chemically match their transported molecule type or *substrate class*, e.g. “hexose sugar” or “hydrophobic amino acid”. On the other hand, the remainder of the sequence, while often structurally conserved within a (super-) family, typically shows low sequence identity even to closely related proteins. This means that by creating a training dataset of transporters that are categorized by their substrate class, we can automatically find statistically significant similarities between proteins transporting the same substrate, and differences to members of another substrate class, allowing us to filter out irrelevant information, and keep those data points that let us accurately tell the classes apart.

On the other hand, there is also the possibility of confounding variables with this approach. Some highly conserved residues are related to the mechanism shared by all members of the protein family, meaning that the model could also encode, for example, the membership in either the MFS or the APC superfamily (see Section 2.1.3.2), in addition to the substrate specificity of the binding pockets. While binding pockets are very common in carrier proteins, they are not always present in ATP-driven complexes, as we noticed for some members of the ABC transporter family, for example. To tackle this problem, we are going to implement a method that finds a set of substrate classes which are well-separable, based on training data. Furthermore, transport protein complexes can undergo gene fusion events throughout evolution, meaning that a bacterial dataset might only contain the transmembrane domains, while a human dataset of the same substrate class contains the full sequence of all domains together. This could potentially be solved by dividing the fused complexes into domains, or by combining bacterial transporter complex operons into one sequence. Finally, the amino acid compositions of β -sheets might be so different from those of α -helical bundles that their encodings are not comparable.

Ion channels achieve their substrate specificity from short sequence motifs that act as *selectivity filters*, containing several ion binding sites that likely aim to re-create the hydration shells of their specific substrate, and/or constrict the channel to only allow certain shapes and sizes of molecules to pass. Whether sequence encodings will be able to capture enough information about the substrate specificity of ion channels, and whether these encodings are comparable to unrelated proteins with ion binding sites, such as APC carriers or P-type ATPases, will have to be tested in practice.

Finally, the training dataset can contain multiple isoforms of one protein, which have very similar sequences to each other, as they are encoded by the same gene, or by very similar structural DNA repeats. This would mean that the dataset contains proteins with very high sequence identity across the entire protein sequence, making it harder to filter out unnecessary data in the training set, and potentially leading to information sharing between training and test sets, which would make the evaluation less reliable. Therefore, it would be a good idea to filter these protein copies out of the dataset with a sequence clustering algorithm.

2.2 Computational tools and data sources

This section covers the data sources and bioinformatics tools, as well as the existing machine learning algorithms and statistical methods used in the remainder of this thesis. Newly implemented algorithms are introduced in the methods section of the chapter where they are implemented and evaluated.

2.2.1 Webservices and databases

2.2.1.1 Biological data and one-letter encodings

The backbone of this project is the vast amount of “-omics” data that is available through various publicly accessible databases. Genomics is the study of the genetic DNA code found in the cell, transcriptomics concerns the RNA transcripts that the DNA can be transcribed into, and proteomics studies the proteins that protein-coding messenger-RNA (mRNA) transcripts are translated to.

Besides those three, some additional terms have been coined, such as *metabolomics* for studying the set of metabolites that is produced in an organism, or *transportome* for the substrates that are transported by the transmembrane transporters present in the cell and its genetic material.

In this thesis, we aim to predict the transportome of the proteome, meaning that we need proteome databases to get the protein sequences, along with tabulated and organized experimental results that tell us which proteins are known to be transmembrane transporters, and which molecules each of them carries across the membrane.

DNA is encoded as single letters that correspond to the four DNA bases *adenine*, *guanine*, *cytosine* and *thymine* (A,G,C,T, respectively), whereas RNA contains uracil instead of thymine, therefore T is replaced by U. Protein sequences are stored in the form of the one-letter amino acid code. Databases typically follow the nomenclature prescribed by the International Union of Pure and Applied Chemistry (IUPAC) [243], which provides single letters for each of the 20 proteinogenic amino acids that are encoded by the standard genetic code (see Figure A.1). Additional codes U and O exist for the amino acids selenocysteine and pyrrolysine, respectively (see Section 2.1.1). Unspecified or unknown amino acids are usually denoted as “X”. A sequence position that is either an Asp or Asn is labeled as “Asx” or “B”. In a similar fashion, Glx (“Z”) can either be Gln or Glu, or any substance that turns into glutamic acid after acid hydrolysis.

Many genes and gene products are known by multiple names and symbols that are subject to change as the scientific knowledge expands. Typically, one primary gene name and one protein name are defined, while the remaining names are treated either as synonyms, or are deprecated if they were wrongfully assigned.

Multiple services exist that maintain primary names for genes, transcripts and proteins. Some act independently of organisms, such as NCBI RefSeq [244] or Uniprot [245]. Others maintain and curate the available data, and assign primary gene names for one specific model organism, such as the Saccharomyces Genome Database (SGD) [246] for *S. cerevisiae*, The Arabidopsis Information

Resource (TAIR) [247] for *A. thaliana*, the HUGO Gene Nomenclature Committee (HGNC) [248] by the Human Genome Organization (HUGO) for human, Mouse Genome Informatics (MGI) [249], the Zebrafish Information Network (ZFIN) [250], and the Vertebrate Gene Nomenclature Committee (VGNC) [251] for vertebrae that do not have their own committee. Some services, such as EcoCyc [252] for *E. coli* or Aramemnon [253] for *A. thaliana* membrane proteins, only curate the available data from other sources, without assigning primary gene or protein names themselves.

Databases also annotate genes and gene products with unique identifiers, in order to simplify automatic processing. These identifiers remain in use across database versions, and follow a standardized pattern. As scientific knowledge expands, some types of identifiers can be merged, split or deleted, and new identifiers can be added. Multiple services exist that provide unique and/or stable identifiers, such as Entrez [254], Ensembl [255], and Uniprot [245]. Update files that list the changes between database versions, as well as conversion and mapping files that allow for the linking of multiple datasets, are available as well [255][248]. Throughout this thesis, we will be using Uniprot accessions to identify proteins, along with the primary gene symbols assigned by Uniprot (if available).

2.2.1.2 Nucleotide databases

Data sources for raw genetics data include the European Nucleotide Archive (ENA) [256] of the European Bioinformatics Institute (EMBL-EBI), the GenBank database [257] maintained by the National Center for Biotechnology Information (NCBI), and the DNA Data Bank of Japan (DDBJ) [258]. Together, they form the International Nucleotide Sequence Database Collaboration (INSDC) project [259] that aims to standardize the storage and processing of data and metadata.

The DNA from an organism is typically assembled into an idealized *reference genome*, which is annotated with genetic variants and mutations found in individual organisms, along with their known effects. The set of variants found in one individual organism is called the *genotype*, and the effect of a particular variant is called a *phenotype*. Various webservices allow the user to interactively browse and/or retrieve reference genomes and genetic variants for any sequenced organism. These include Ensembl [260], the UCSC genome browser [261], as well as several NCBI services, such as RefSeq [244], Entrez [254], Clusters of Orthologous Genes (COGs) [262], and the Genome Data Viewer (GDV) [263].

Besides genetic variants, the reference genome is also annotated with regulatory features that affect the gene expression, and with the different RNA transcripts that a particular gene can produce. The DNA (and RNA) that encodes the amino acids of a protein is called a coding sequence (CDS), which is, in turn, the basis for the reference proteome.

2.2.1.3 Uniprot

Uniprot is a collection of multiple protein databases, along with a webservice for retrieving protein sequences, annotated data, and links to related databases [245].

The Uniprot Knowledgebase (UniprotKB) is a protein database mainly based on translated CDS data from the three INSDC nucleotide archives, which makes up around 95% of the entries. The remaining 5% stem from other sources, such as the protein 3D structure database *Protein Data Bank* (PDB) [264], from direct protein sequencing, or from scanned literature. Researchers can also submit their protein sequences directly to the database [265][245].

The UniprotKB is split into two parts: The Swiss-Prot database contains entries that are manually reviewed by researchers, whereas the proteins in TrEMBL are automatically added and annotated from outside sources. An entry in the database does not just contain the sequence, but also additional information that is available about the protein, including its functions and locations in the cell, its predicted and/or experimentally derived 3D structures, membership in protein families, temporary or permanent interactions with other proteins in order to form quaternary structures, and links to other databases. These annotations are often backed up by experimental evidence from literature, either direct evidence that concerns the exact protein, or information that was inferred from a homologous protein of sufficient similarity [245].

A *canonical sequence* is selected as a representative for each protein, based on functionality, gene expression, conservation and transcript length. Isoforms that are encoded by the same gene in the same species (e.g. splice variants) are either part of the same entry, but they can also be listed as separate proteins [266]. In the latter case, they might have to be removed from the dataset when evaluating predictive models, as they could lead to information sharing between training- and test set.

Each entry also contains information about whether the protein encoded by a CDS has been experimentally observed *in vivo*, and whether the gene that encodes the protein is known. When training machine learning models, this can be seen as a trade-off between higher data quality/reliability on one side, or a higher number of available samples per class on the other (see e.g. Appendix C.1 from Chapter 4).

As there is a large amount of redundant information available across publicly accessible protein databases, the *Uniref* datasets [267] aim to cluster proteins at different levels of sequence identity (i.e. 50%, 90% or 100%), to then pick a representative protein for each cluster. Representatives are selected according to measures related to biological relevance and data quality, as well as the sequence length [268]. Uniref is based on a subset of UniParc [269], which is a non-redundant protein database that contains sequences from a wide range publicly available sources, including UniProtKB, RefSeq, Ensembl, and many more. Applying an algorithm to only the representatives instead of the entire dataset allows the user to remove a large amount of redundancy, without losing too much biological information. In the following chapters, we will use Uniref

clusters to calculate conservation scores for the 20 amino acid types in each protein.

2.2.1.4 Protein Data Bank

The Protein Data Bank (PDB) [264] is a database of three-dimensional protein structures and complexes, often together with bound ligands and substrates. These structures are provided as PDB files, where the 3D position of each atom is listed, along with different types of bonds, secondary structures that are formed, and the resolution (in Ångström) at which the structure was experimentally determined. Most entries are annotated with the peer-reviewed study where they were published, but it is also possible to submit structures directly without an associated publication [264]. The PDB is managed by the worldwide PDB (wwPDB) organization, which is a collaboration between the Research Collaboratory for Structural Bioinformatics (RCSB), the European Bioinformatics Institute (EBI), and the Protein Data Bank Japan (PDBj) [270].

PDB entries are determined through various experimental methods, such as X-ray crystallography, Nuclear Magnetic Resonance (NMR) spectrometry and cryogenic electron microscopy (cryo-EM) [271]. The Computed Structure Models (CSM) archive [272] expands this structure repository by computationally predicted structures from ModelArchive [273] and AlphaFoldDB [274][21][275].

There are multiple webservices that provide convenient access to the PDB, including the RCSB.org website, the Protein Data Bank in Europe (PDBe.org), and the Protein Data Bank Japan (PDBj.org) [270]. Some of these services also include other tools, such as Mol* [129] or JSmol [276] for visualization, MemProtMD [277] or PDBTM [278] to determine the membrane orientation of transmembrane proteins, or a sequence annotation viewer for finding annotated domains and motifs [272]. We used RCSB.org together with Mol* and MemProtMD/PDBTM to create most of the rendered protein structures in Section 2.1.

2.2.1.5 Ontologies and gene/protein annotations

In order to efficiently take advantage of the existing scientific research literature, there are several ways in which gene and protein sequences are annotated with additional knowledge. If an experimental study found new information about a protein, e.g. that it is responsible for a particular task in the cell, associated with a known pathway, or located in only a subset of cellular components, then the corresponding gene or protein identifiers are linked to a gene annotation, along with the study or studies that support this information.

One of the most commonly used annotation systems is the Gene Ontology (GO) [47]. The GO consortium provides three ordered sub-hierarchies (i.e. three *aspects*) of GO terms: Molecular Function (MF), Biological Process (BP), and Cellular Component (CC). A molecular function directly describes a task that a protein carries out in the cell, while BP describes a pathway or broader goal that the proteins molecular function contributes to. Each entry has a trivial name and an identifier (e.g. “transmembrane transport” is “GO:0055085”). A GO term can have multiple synonymous names and identifiers (IDs), although

one of each is always selected as the primary name and ID. Terms can be added, deprecated, split or merged between different versions of GO.

GO terms are connected by directed edges, which are in turn annotated with the type of relation that exists between the two GO terms on either side of the edge. A child term is connected to a parent term with an edge that points from the child towards the parent. Each edge is also annotated with the type of relation it represents, e.g. “part_of”, “regulates” or “is_a”. For example, the GO term *transmembrane transporter activity* “is_a” *transporter activity*, and that “is_a” *molecular function*, which is in turn the root node of the MF aspect [279]. These edges can also exist between aspects, e.g. *protein transport within plasma membrane* (BP) “occurs_in” *plasma membrane* (CC), and *carbohydrate transmembrane transporter activity* (MF) is “part_of” *carbohydrate transmembrane transport* (BP). Generally, the “is_a” relation is only used between two GO terms from the same aspect, as it describes a hierarchical relationship where one GO term is a sub-type of another. This means that every protein that is annotated with the more specific term is also implicitly annotated with the more abstract term. The sub-tree of GO terms that has a path to a particular node is also referred to as its *descendants*, and the path between a GO term and the root node of its aspect makes up the set of its *ancestors*.

Ontologies like GO can be represented in the human-readable, text-based Open Biological and Biomedical Ontologies (OBO) [280] file format, which was developed specifically for bio-ontologies. The OBO foundry [280] standardizes the definitions of abstract terms and the types of relations that can occur between different entities across different ontologies, through the Core Ontology for Biology and Biomedicine (COB) and the Relation Ontology (RO), respectively. The OBO format is related in concept to the much more complex and more machine-oriented Web Ontology Language (OWL) [281]. OBO can be converted to OWL to get access to more sophisticated constructs that allow for additional workflows, e.g. for automated logical reasoning, or cross-referencing between ontologies [282][283]. Most available tools and services for creating, maintaining, searching and cross-linking bio-ontologies either use OWL internally, or are built on OWL entirely, due to its more advanced capabilities [280][284][285][286][287]. These datasets can usually be converted to and from OBO files, in order to take advantage of the simpler and more human-readable/writable format.

Protein and gene databases, such as Uniprot, then use this ordered hierarchy of GO terms to create GO annotation (GOA) datasets [288]. Webservices like QuickGO [289] and AmiGO [290] allow for easy access to the GO annotations that are provided by various protein databases. These relations between proteins and GO terms are labeled with qualifiers that describe the type of relation. For example, the human sugar uniporter GLUT3 (see Section 2.1.3.2) “enables” the molecular function *D-glucose transmembrane transporter activity*, and is “located_in” the *plasma membrane* (CC) [279]. Qualifiers can also be negated with the “NOT” prefix, e.g. human GLUT3 explicitly “NOT|enables” *fructose transmembrane transporter activity* (MF). Besides the qualifier, each annotation is also labeled with a reference, e.g. with a link to a published research paper or to a protein family database. This reference is accompanied with an evidence code that describes whether the experiment in the study was carried out on the exact same protein or on a similar protein, and whether the annotation was manually

checked by a researcher. Filtering out annotations that were automatically inferred can therefore increase the data quality, while also decreasing the number of available annotations.

Another set of protein annotations is provided by Uniprot in the form of Uniprot Keywords [291]. Keywords are a controlled vocabulary with a hierarchical structure, but they lack the formal relationships between terms, such as “is_a” or “part_of”, meaning they do not represent a full ontology like GO, and do not directly support associated OBO or OWL workflows. They are, however, linked to synonymous GO terms. Compared to GO, the keyword dataset is much smaller and the information provided by annotations is not as detailed as many of the more specific GO terms. In the Swiss-Prot subset of Uniprot, the keywords are assigned manually by curators [291].

The ChEBI ontology [292] contains “Chemical Entities of Biological Interest”, i.e. specific molecules that occur in and around living organisms, as well as associated abstract descriptions. For example, the molecule *D-glucopyranose* “is_a” *D-glucose*, which “is_a” *glucose* [293]. The root node of the ontology that all molecules are descendants of is called *chemical entity*. The ChEBI ontology is cross-referenced to the Gene Ontology, which can be useful for connecting a molecular function GO term annotated to a protein in our dataset to its transported substrate in the ChEBI ontology. The GO-MF term *D-glucose transmembrane transporter activity* “has_primary_input” *D-glucopyranose* (ChEBI) [279], for example. These cross-references can be retrieved from webservice such as QuickGO [289] and Ontology Lookup Service (OLS) [285], or calculated manually from OWL files, with software like Protégé [286] or ROBOT [284].

2.2.1.6 Protein families, domains, motifs and sites

There are many different databases for protein sequence annotations, most of which are accessible through InterPro [294]. This resource links the protein sequences from the UniprotKB database to their families, superfamilies, domains, repeats and functional sites, and visualizes where the annotated residues are located within the sequence.

Protein families are groups of proteins that are related through evolution and share a common ancestor, which is typically reflected by similarity and conservation of their sequences and/or 3D structures, and often through similar functions. They tend to show relatively high sequence similarity, while the structural fold can differ between members. *Homologous superfamilies* on the other hand, such as MFS and APC from Section 2.1.3.2, show a lower sequence similarity, have a more distant common ancestor, and are often defined by having a similar structural fold. A *protein domain* is a part of a protein that folds independently, and often fulfills a particular function, e.g. the ATP binding domain(s) in primary active transporters (see Section 2.1.3.1). Sequence motifs, also called *repeats*, are short sections of an amino acid sequence that show conservation across evolution, e.g. the *sugar porter motif* from Section 2.1.3.2. Finally, a *functional site* is directly involved in the function of the protein, such as substrate binding sites (e.g. the residues in the sugar binding pocket from Section 2.1.3.2, which are listed in Figure A.3), active sites in enzymes, residues

where post-translational modifications (PTM) take place, and other types of conserved sites [294].

These InterPro entries are manually mapped to matching GO terms by curators, based on biological knowledge and scientific literature. The proteins associated with these InterPro entries are then automatically annotated with the corresponding GO terms. InterProScan [295] can search the sequences in a protein database for different InterPro types (e.g. for domains or repeats) using predictive models provided by member databases, and can thereby provide a first estimate of their functions, if matching entries are found.

2.2.1.7 Transmembrane transporter databases

Besides the Gene Ontology, UniprotKB and InterPro, there are also more specialized gene and protein annotation and classification resources that focus specifically on transmembrane transport proteins.

The Transporter Classification Database (TCDB) by Saier Jr et al. [118] hierarchically classifies transporters according to their mechanisms, superfamilies and phylogenetic relationships. In the TC system, each protein is assigned an identifier V.W.X.Y.Z, where V and W refer to the class (e.g. primary active, carrier, channel) and subclass, X and Y can correspond to the superfamily, family or sub-family, and Z represents the substrate(s) of the protein. In other words, the TC system classifies membrane transporters by their transport mechanism first, by evolutionary relationships second, and by substrates third, meaning that it is more focused on similarities in protein structures and/or sequences, and less on substrate binding pockets or selectivity filters. Most nodes in the hierarchy are annotated with a text about the current state of the field, along with supporting research papers. Mapping files to UniprotKB, GO, ChEBI, PDB, RefSeq and protein family databases are provided as well [296][233].

TransportDB [297] is a database of predicted membrane transporters that is based on the TransAAP pipeline [298] for automatically assigning TCDB identifiers to previously unclassified transporters in prokaryotic genomes. First, it uses BlastP [299] to search for proteins with homologous sequences in RefSeq-bacteria [244], COG [262] and TCDB. The pipeline then employs hidden markov models in HMMER3 [300] to find parts of sequences that resemble known domains and motifs, as well as TMHMM [99] to predict transmembrane helices. While the first version required manual curation, version 2.0 aims to automate that process, and uses a combination of PHP and Perl scripts to improve the results through empirically derived rules. Some eukaryotic genomes are also included, although their annotation still requires extensive manual curation, due to the increased complexity of eukaryotic genomes.

The HGNC [301], along with other mammalian gene naming committees mentioned in Section 2.2.1.1, maintains systems of standardized names for genes of human transporters and channels, based on their mechanism, subtype, substrates and sequence similarity, respectively.

The Solute Carrier (SLC) system [158] divides 473 human carrier protein genes (as described in Section 2.1.3.2) into 76 different families [302]. For example, the aforementioned GLUT protein family of glucose uniporters is encoded by

the genes that make up the SLC2 gene family, with the GLUT3 gene listed as SLC2A3 [303]. Orthologues in a select group of mammals, such as mouse and rat, are also named according to the SLC system, whereas homologues in other types organisms, e.g. in bacteria, are not included.

A similar naming convention exists for human (and some mammalian) ABC proteins and complexes (see Section 2.1.3.1), with 51 human genes [304] divided into seven subfamilies ABCA-ABCG that were originally based only on sequence similarity [142], while family membership is now also determined by substrates, and by domain architecture (e.g. full- and/or half-transporters) [143]. Only five of those seven families contain genes that encode transporters, as ABCE and ABCF family members lack the transmembrane domains (TMDs) [143].

The 142 human ATPase genes are divided into eleven families and sub-families that encompass all genes needed to assemble the complexes. This means that many of the genes with the corresponding “ATP” prefix do not directly interact with the substrate in the membrane. Some complexes classified as ATPases do not act as membrane transporters and/or are do not possess integral transmembrane domains [305].

Ion channels and aquaporins (see Section 2.1.3.5) are named according to their substrate and/or gating mechanism. KCN, SCN and CACNA are prefixes for channels that carry potassium, sodium and calcium, respectively; AQP is the prefix for aquaporins. Most of these channel families also have sub-families, for example the potassium channels from Section 2.1.3.5 (KCNA1 and KCNA2) are members of the potassium voltage-gated channel subfamily A. There are 330 human genes listed that code for ion channels, including aquaporins [306].

A database for clinically and pharmaceutically relevant ion channels and transporters is provided by the Guide to Pharmacology database (GtoPdb) [307]. It classifies channels and aquaporins by their gating mechanism (ligand-gated, voltage-gated, other) and their substrates, whereas transporters are categorized into ABC family members, P-type ATPases, F/V-type (rotary) ATPases, and members of the SLC superfamily. It focuses on human proteins and complexes, but also lists transporters from other species when they are clinically relevant (e.g. bacterial transport proteins implied antibiotic resistance), or when they are relevant to comparisons between human and model organisms used in drug testing, such as mice and rats.

Finally, drug target databases such as DrugBank [308] list all known drugs and xenobiotics that are transported by membrane proteins acting as drug targets, e.g. the multi-drug ABC transporters P-glycoprotein (ABCB1) and breast cancer resistance protein (ABCG2).

2.2.1.8 BLAST and position-specific scoring matrices

Basic Local Alignment Search Tool (BLAST) [24][299] is a software that allows for the search of protein (BLASTP) or nucleotide sequences (BLASTN) in corresponding databases. It divides the sequence into *k-mers*, i.e. into sub-sequences of length *k*, and then searches for matching *k-mers* in sequences from the database. When a match is found, it uses the *k-mer* as a starting point (seed) to build up a *local alignment*. To account for insertions and deletions

of amino acids during evolution, BLAST also allows for gaps to occur in the alignment that represent insertions or deletions of amino acids at a particular position. In that case, one or multiple sequence positions (i.e. amino acids or nucleotides) in one of the sequences is matched to the symbol “-” instead of an amino acid. Figure A.4 shows an example of a pairwise sequence alignment.

Protein sequences are often compared by employing *substitution matrices*. These 20×20 integer matrices contain empirically derived log-odds values, where positive values indicate that the substitution of one particular amino acid type (see Figure A.1) by another throughout evolution is more likely than random, whereas a negative value means that it is less likely. A log-odds score is the ratio between an observed and an expected probability, brought to a linear scale that is centered on 0, by applying a logarithm:

$$\text{log-odds} = \log \left(\frac{P(\text{event})}{P(\text{background})} \right) \quad (2.1)$$

Following this formula, the substitution score for amino acid types a and b are calculated as:

$$s(a, b) = \log_2 \left(\frac{p(a, b)}{q(a) \cdot q(b)} \right) \quad (2.2)$$

, where $p(a, b)$ is the observed frequency with which a and b are paired in an alignment, and $q(a)$ denotes the relative amino acid frequency of residue a across all sequences in the dataset. The product $q(a) \cdot q(b)$ is the expected probability of this pairing occurring by chance. Examples for amino acid substitution matrices include the *Point Accepted Mutation* (PAM) [309] and *Blocks Substitution Matrix* (BLOSUM) [68] matrices, which differ in the sequence dataset they are based on. PAM is derived from global alignments of homologous protein sequences, whereas BLOSUM is calculated from local sequence alignments of conserved sequence motifs (called *blocks*). There are multiple versions of each matrix that are built from more closely or distantly related proteins. The most commonly used substitution matrix, and default parameter in many alignment tools, is the BLOSUM62 matrix [310], where scores are calculated based on blocks that were clustered with a 62% sequence identity threshold to reduce redundancy (see Section 2.2.1.9).

The problem of finding pairwise alignments can be optimally solved by the Needleman-Wunsch [311] and Smith-Waterman [312] algorithms that find the highest-scoring global or local alignment, respectively. They create a $(n+1) \times (m+1)$ matrix for the pairwise alignment of two sequences with lengths n and m , which is subsequently filled by a dynamic programming algorithm that considers all possible alignments, and scores them according to log-odds values from substitution matrices, together with an additional gap penalty for adding insertions or deletions to the pairwise alignment. While optimized versions of the algorithms exist that can reduce the space requirements to $O(\min(m, n))$ [313], the runtime of $O(m \times n)$ for each sequence in the database is computationally very intensive when considering the sizes of sequence databases. The default database used by BLASTP (the NCBI non-redundant protein database *nr*) contains 894,448,555 sequences at the time of writing [314].

Instead of calculating an optimal alignment for each sequence, BLAST creates an index of k-mers that is similar in principle to a hash-map, which speeds up the

search of local alignment starting points (i.e. *seeds*) significantly. It then extends the seed to surrounding sequence positions, and directly stops the extension when its raw alignment score, which is calculated by simply adding together the substitution matrix entries and gap penalties for the aligned positions, does not reach a minimum threshold (a higher score is better). Finally, matching sequences in the database are ranked by a normalized score, the *E-value*, which denotes the statistical significance of the result based on the Karlin-Altschul statistics [315] (lower is more significant). It is defined as the number of sequence alignments with a score at least as good or better that one would expect to find randomly in the given BLAST database, given the length of the input sequence and selected substitution matrix [24].

The protein sequences with an E-value below a given threshold (e.g. $E \leq 0.002$) are then returned as the result. Position-specific iterated BLAST (PSIBLAST) [299], an extension of BLASTP, takes these proteins, calculates a multiple sequence alignment (MSA) between them (see Figure A.3), and then uses that to derive a new substitution matrix. The difference is that this *position-specific scoring matrix* (PSSM) is calculated for every position in the input sequence, not just for each type of amino acid. This PSSM is then used as the log-odds scoring matrix in another search iteration in the same database. By using a score that is specific for the protein and its sequences, instead of a generalized substitution matrix for all proteins, PSIBLAST can find even more distantly related homologues in each iteration. This is useful for finding proteins that are related and have a similar function, but are not obviously similar in their protein sequences. As discussed previously, this is the case for many transmembrane transport proteins, e.g. for those in the MFS or APC superfamilies from Section 2.1.3.2.

Supervised machine learning models need a protein to be encoded by a fixed-size vector, whereas protein sequences (and structures) in nature have variable lengths and sizes. This means that they need to be encoded by an algorithm. Calculating a PSSM after n iterations of PSIBLAST on one of the clustered Uniref databases introduced in Section 2.2.1.3, and then averaging the scores of each amino acid type across the sequence positions, allows us to create a specific substitution matrix for each individual protein. The values in the 20×20 matrix correspond to the likelihood that a particular amino acid type in the protein is exchanged by another through evolution. In the first half of this chapter (see Section 2.1.3.7), we learned that the types of amino acids that are over-represented and/or conserved in a protein are in many cases directly related to the specific function that the protein carries out, e.g. by guiding, selecting and binding the substrate into a transmembrane domain, or by enabling the mechanism that transports the molecule to the other side of the membrane, through a series of conformational changes in the protein structure. In the following chapters, we will introduce a machine learning feature calculated from multiple standardized PSSM features for the same protein (Multi-PSSM) that includes information derived from very closely- and very distantly related proteins by using multiple sequence databases and changing PSIBLAST parameters.

Calculating multiple PSSMs for each protein in our dataset is made possible by the software package *blast+* [316] that allows for the local usage of these tools. The *makeblastdb* utility converts a FASTA file containing text-based

sequence data into a pre-processed BLAST database file that contains data structures which speed up the sequence searches. Tools to search for matching proteins in a database, such as *blastp*, are included as well, and support efficient multi-threading. The *psiblast* command line program can carry out one or multiple iterations automatically, and provide the user with a PSSM derived from the results of the last iteration. We implemented a Python wrapper that automatically creates PSSMs for a protein dataset, and supports caching (see Appendix C.1).

2.2.1.9 Sequence clustering methods

In order to reduce redundancy in protein sequence datasets, and counteract the possibility of information sharing between training- and evaluation datasets used for performance testing of predictive models, sequences can be clustered according to a particular identity threshold. If more than a given percentage of positions is identical between two proteins, then both sequences are assigned to the same cluster. A *representative* can then be selected for each clustered group based on various criteria. Applying a software (such as BLASTP from the previous section) to only the cluster representatives instead of the entire, unclustered database, can speed up its runtime significantly.

CD-HIT [317] is a commonly used tool for clustering protein sequences. It speeds up the clustering process by employing a *short word filter*, which can estimate the sequence identity between two protein sequences from the number of identical k-mers between the two sequences, without performing an actual alignment. If the pairwise identity can not be confidently estimated with the filter, an alignment is performed instead. The greedy, incremental clustering algorithm first sorts proteins according to their sequence length, and the longest sequence is selected as the representative of the first cluster. The remaining proteins are then compared to the existing representatives. If the identity of a protein to any of the cluster representatives is above the given threshold then they are assigned to the corresponding cluster, otherwise they become the representative of a new one.

As mentioned before in Sections 2.2.1.3 and 2.2.1.2, databases of pre-calculated protein and gene clusters exist as well. Some databases, such as Clusters of Orthologous Groups (COGs) and eggNOG [318], employ clustering methods across a wide range of available genomes to find orthologs, i.e. homologs that are evolutionary descendants of the same ancestor gene (as opposed to paralogs, which are homologs created by gene duplication events that then evolved separately). Uniref [267] starts with a very large, non-redundant protein dataset that contains data from multiple sources, and assigns the proteins into clusters of 100%, 90% or 50% sequence identity. Proteins can only be assigned to a cluster if they have an overlap of at least 80% with the longest sequence, which is called the seed [268]; sequences with ten or fewer residues are only included in Uniref100, as they can cause problems with the greedy clustering algorithm [319]. Representatives for the clusters are selected based on biological relevance, availability and quality of annotation data, as well as sequence length [320][268].

2.2.2 Statistical methods

Throughout this thesis, we use statistical methods to test our hypotheses, for example to learn about the distribution of a variable, find statistically significant differences between two groups of samples, or select those data points that have the highest predictive value in a classification task. In this context, the *null hypothesis* H_0 denotes the default assumption that there is no effect, e.g. that there is no difference between the two groups of samples, and the *alternative hypothesis* H_1 represents the opposite. The goal of hypothesis testing is then to find enough evidence to *reject* H_0 , which in turn supports H_1 [321]. The null hypothesis is usually rejected based on the *p-value* and/or the *test statistic*, which are provided by the statistical test. The p-value is commonly defined as the probability of observing the given test results, under the assumption that H_0 is true [322]. If this probability is equal or lower than the *significance level* α , i.e. if $p \leq \alpha$, with $\alpha \in [0, 1]$, then the H_0 is rejected, and the experimental result is *statistically significant*. The most commonly used value for α is 0.05 [323].

Wrongfully rejecting H_0 when it is true is called a Type I error, whereas failing to reject a false null hypothesis is referred to as a Type II error [324]. In machine learning, these are also called false positives (FP) and false negatives (FN), respectively. A true negative (TN) is defined accordingly as not rejecting a true H_0 , while correctly rejecting a false null hypothesis produces a true positive (TP) [325].

2.2.2.1 Multiple hypothesis correction

When performing n hypothesis tests at a significance level of α then the expected number of false positives purely by chance is equal to $n \times \alpha$, which increases the Type I error rate. Multiple hypothesis correction adjusts the p-values derived from the tests, in order to account for this problem.

Bonferroni correction [326] is a rather conservative method, meaning it leads to a lower probability of Type I errors, but increases the risk of Type II errors (false negatives). The corrected p-value p' is calculated as:

$$p' = p \times n$$

This increases the p-values by a factor of n , meaning that fewer might fall below the significance threshold α , causing the rejection of their associated null hypothesis.

A less conservative approach for multiple hypothesis correction is the Benjamini-Hochberg (BH) method [46]. Given n hypothesis tests, it first sorts the n p-values in increasing order and assigns ranks based on the position in the sorted list:

$$p_1 \leq p_2 \leq \dots \leq p_n$$

The method then finds the largest rank $i \in \mathbb{N}, 1 \leq i \leq n$ where

$$p_i \leq \frac{i}{n} \times \alpha$$

and finally rejects all null hypotheses where $p \leq p_i$.

Whereas Bonferroni controls the probability of *any* FP occurring, BH keeps control over the *expected proportion* of rejected null hypotheses that are actually false positives, which is also referred to as the False Discovery Rate (FDR).

2.2.2.2 The normal distribution, and testing for normality

Some statistical methods, such as the independent t-test described in the next section, assume that the data is (at least approximately) distributed according to the *normal distribution*, and breaking that assumption can lead to unreliable or misleading results.

A perfect, idealized normal distribution is defined by its characteristic bell shape that is symmetrical around the mean value μ , which is equal to the median of the distribution (i.e. the 50th percentile, where 50% of samples have higher values), and to the mode (the most common value). Due to the symmetry around the mean, the tail of the distribution can be defined by a single parameter, the variance σ^2 , which is the square of the *standard deviation* σ . A higher σ denotes a larger spread of the values [323].

If mean and variance are 0 and 1, respectively, then that is referred to as the *standard normal distribution*. A realization x of a normally distributed variable $X \sim \mathcal{N}(\mu, \sigma^2)$ can be *standardized* by applying the z-score [324]:

$$z = \frac{x - \mu}{\sigma}$$

The Shapiro-Wilk test [327] is a statistical test for the null hypothesis that a given sample was drawn from a normally distributed population. It produces a p-value, along with test statistic $W \in [0, 1]$, where higher values indicate closeness of the variable to a normal distribution. The null hypothesis is rejected if W is lower than a pre-defined *critical value*, or if $p \leq \alpha$ for significance level α . A rejection of H_0 is an indication that statistical tests that assume normality of the data should not be applied to the given set of samples. Whereas the initial version of the test was restricted to $n < 50$ samples, Royston [328] created a modified version that works for $n < 2000$, and then further adjusted it for $3 \leq n \leq 5000$ [329].

When comparing eight statistical tests for normality, Yap and Sim [330] found that Shapiro-Wilk (SW) was the most powerful test overall, and was particularly strong for small sample sizes. Razali et al. [331] came to the same conclusion when comparing SW to three commonly used tests for normal distribution.

2.2.2.3 Testing for differences of the mean between two groups

The *two-sample* t-test is a statistical test that aims to compare the means of two groups of samples. Preconditions are that the two groups are independent of each other, and that the data in each group is normally distributed. As for requirements around the variances of the groups, two different versions exist: The Student's t-test [332], also called the *pooled t-test*, assumes equal variance, whereas Welch's t-test [333] assumes the variances to be unequal.

The test statistic t for Welch's t-test is calculated as:

$$t = \frac{\bar{x}_1 - \bar{x}_2}{\sqrt{\frac{s_1^2}{n_1} + \frac{s_2^2}{n_2}}}$$

, where $\bar{x}_i$ denotes the mean, s_i the variance, and n_i the number of samples in group $i \in 1, 2$ [334].

The formula for the test statistic t standardizes the observed difference in the means of the groups (the numerator), by dividing it by its estimated standard deviation (i.e. the standard error). The absolute value of t can now tell us how many standard deviation units the difference is away from 0. The null hypothesis H_0 is that the *population means* μ_1 and μ_2 of groups 1 and 2 are equal, i.e. that $\mu_1 = \mu_2$, while H_1 is that $\mu_1 \neq \mu_2$. The null hypothesis is rejected when the absolute value of t exceeds a critical value t^* . This constant is calculated from the degrees of freedom v and the significance level α , using the underlying *Student's t-distribution*. The value of v for Welch's t-test is, in turn, approximated from the sample sizes and standard deviations of the two groups [334].

In contrast to this two-tailed t-test that tests for equal or different population means by considering the absolute value of t , the one-tailed two-sample t-test only considers one side of the distribution in order to test the alternate hypothesis that one of the means is greater than the other. In this case, the null hypothesis is still $H_0 : \mu_1 = \mu_2$, while the alternate hypothesis is either $H_1 : \mu_1 < \mu_2$ or $H_1 : \mu_1 > \mu_2$. This means that the direction (greater or smaller) has to be known beforehand, as the test does not consider the other direction. To account for the two sides in a two-tailed test, the value of the used significance level α has to be divided by two before retrieving the value from the corresponding table or software, which is not the case for one-tailed tests [334].

Finally, the *one-sample* t-test checks for the alternative hypothesis that the true (typically unknown) mean value μ of a population of samples is higher than, lower than or unequal from the hypothesized population mean μ_0 , for the right-tailed ($H_1 : \mu > \mu_0$), left-tailed ($H_1 : \mu < \mu_0$) and two-tailed ($H_1 : \mu \neq \mu_0$) variants, respectively, while the null hypothesis is denoted as $H_0 : \mu = \mu_0$. The test statistic is given as:

$$t = \frac{\bar{x} - \mu_0}{\frac{s}{\sqrt{n}}}$$

, where $\bar{x}$ is the observed mean of the sample group, n is the size of the group, s is its standard deviation, and μ_0 is the assumed mean of the underlying population. The absolute value of the numerator becomes larger when the observed mean $\bar{x}$ diverges more from the hypothesized mean μ_0 , which is then standardized by dividing through the standard error of the sample group, such that t is on a scale of standard deviation units [334].

2.2.2.4 The ANOVA F-score

The one-way analysis of variance (ANOVA) F-test is a parametric test that can find statistically significant differences between the means of more than

two groups. We are going to use the F-score throughout this thesis to find and remove variables from predictive models that do not show a significant difference between two or more classes that the samples are assigned to [334]. For example, we will encode proteins belonging to two or more classes (e.g. *sugar carrier* and *amino acid carrier*) into fixed-length numeric vectors, and then remove positions from these vectors that give the least amount of information about which of the classes a protein is assigned to, based on training data (see Section 2.2.3.2). This requires the meaning of a particular vector position i to remain constant across all proteins. For example, the sequence position 1 could always contain the normalized frequency of the amino acid *alanine* (see Figure A.1) in the protein sequence, and we want to know if this variable shows significant differences in the mean between different groups of transmembrane transport proteins.

The null hypothesis of the ANOVA F-test assumes that all m population means are equal, whereas the alternative hypothesis is that *any* pair of means is different [334]:

$$\begin{aligned} H_0 : \mu_1 &= \mu_2 = \dots = \mu_m \\ H_1 : \exists i, j \in \{1, 2, \dots, m\} : \mu_i &\neq \mu_j \end{aligned}$$

As with the other statistical tests, the F-test produces a test statistic F , which can reach or exceed a critical value F^* , at which point the result is considered to be statistically significant, meaning that the null hypothesis is rejected. This critical value is, in turn, calculated from the F-distribution, with significance level α and the two degrees of freedom $df_1 = I - 1, df_2 = N - I$ as parameters, where I is the number of groups and N is the total number of observations across all groups. F itself is calculated by dividing the variations in the sample means by the variation among individuals in the same sample [334].

Conditions for applying ANOVA include that the variables follow a normal distribution, and that they have the same standard deviation, which can be unknown [334].

2.2.2.5 Non-parametric tests

Tests that do not assume a specific distribution of the data are called non-parametric statistical tests. This is useful, for example, in cases where we want to perform a two-sample t-test, but the Shapiro-Wilk test tells us that the requirement of normality is not fulfilled. Other solutions to that problem can be data transformations, removal of outliers that stem from different populations, or permutation testing, among others [334].

A non-parametric replacement for the two-sample t-test is the *Wilcoxon rank sum test* [335], which is also called the *Mann-Whitney U test* in some statistics software packages, like SciPy [336]. Wilcoxon and Mann-Whitney use different test statistics W and U , respectively, that are different formulations of the same concept, and always produce the same p-value [334].

As the name suggests, the Wilcoxon rank sum test uses ranks instead of raw values. The requirement of independence between samples in the groups also

applies to this test, meaning that samples should not be paired between the groups (e.g. two sets of blood samples taken from the same group of patients at two different times, in a clinical study). The null hypothesis is that the samples in distinct groups 1 and 2 follow the same distribution. Accordingly, the alternative hypothesis for the two-tailed variant is that the distributions are not the same, while the one-sided variants test if group 1 contains, on average, either larger or smaller values than group 2 [334].

The number of samples in groups 1 and 2 are denoted as n_1 and n_2 , respectively. The test first sorts the $N = n_1 + n_2$ combined data points from small to large, and then assigns a rank $r \in \{1, \dots, N\}$ to each. If multiple samples have the same value then their ranks are averaged. The sum of all ranks of samples that came from the first sample group is now the test statistic W . If ranks were *randomly* assigned, then the average rank would be:

$$\bar{r} = \frac{1 + \dots + N}{N} = \frac{N(N+1)/2}{N} = \frac{N+1}{2}$$

Following from that, the expected mean value μ_W of the sum W would be:

$$\mu_W = n_1 \times \bar{r} = \frac{n_1(N+1)}{2}$$

The standard deviation (following a different logic) is calculated as:

$$\sigma_W = \sqrt{\frac{n_1 n_2 (N+1)}{12}}$$

Since the distribution of W approaches a normal distribution $\mathcal{N}(\mu, \sigma)$ as the sample size increases [334], we can now standardize W , and determine how many standard deviations the observed rank sum W is away from the expected mean μ_W . A sufficient distance can then be used as evidence to reject H_0 . The associated p-value can be approximated from the standard normal distribution, after standardizing W by applying the z-score

$$z = \frac{W - \mu_W}{\sigma_W}$$

, usually together with *continuity correction*, which improves the accuracy of approximating the *discrete* W statistic with the *continuous* normal distribution [334]. If group sizes are small, and no duplicate values that share the same rank are present, then the exact distribution of W , which is calculated from all possible combinations of ranks, is typically used to find a p-value, instead of the normal approximation [337].

2.2.2.6 Correlation coefficients

Pearson's correlation coefficient $r \in [-1, 1]$ is calculated as:

$$r = \frac{\sum_{i=1}^N (x_i - \bar{x})(y_i - \bar{y})}{(N-1)s_x s_y}$$

, from N pairs of values $(x_i, y_i), i \in \{1, \dots, N\}$ belonging to the variables x and y , where $\bar{x}$ and $\bar{y}$ are the means, and s_x, s_y are the standard deviations of the two sample groups, respectively [323].

It is a measure for the linearity of the relationship between two continuous variables x and y . If the relationship is perfectly linear, meaning that the values of x and y form a straight line, then r is either 1 if x increases as y increases, or -1 in cases where x increases while y decreases. If one variable always remains constant while the other changes then their correlation is undefined. In other cases, the correlation coefficient takes on values in $(-1, 1)$. A value of $r \approx 0$ indicates that there is no linear relationship between x and y [338].

However, there could also be a non-linear relationship between x and y , which can be found with rank-based correlation coefficients, such as Spearman's rank correlation ρ [339]. Spearman's method sorts the values of x and y from small to large and assigns integer ranks to them, starting from 1. By calculating the Pearson correlation of these ranks, it can detect ascending or descending monotonic relationships between two variables, which leads to positive and negative values of ρ , respectively. A value close to 0 indicates that no monotonic association was found. Spearman's rank is also applicable to ordinal variables, whereas Pearson's r requires x and y to be continuous. Furthermore, rank-based correlation coefficients are more robust against outliers in the data [338][323].

Correlation analysis can be useful as a starting point for further testing, but does not imply a causal relationship between the two variables, as a high correlation can occur by chance, or it might be a result of an additional, potentially unknown variable [338].

2.2.3 Machine learning methods

Throughout this thesis, we will be using algorithms that encode biological data into numerical feature vectors, so that machine learning algorithms can learn from them. These machine learning methods include unsupervised learning, such as clustering, as well as supervised methods, in the form of classifiers. Supervised learning means that there is a training dataset available where n samples (in the form of d -dimensional feature vectors) were previously assigned with *class labels*, e.g. a dataset of encoded transport proteins with known substrates from annotation data. A supervised learning algorithm then learns to associate the input values with the output, in order to predict class labels for samples where class membership is unknown. Unsupervised learning methods, on the other hand, group the samples based on the feature data itself, typically through algorithms that are based on pairwise similarity measures, meaning that no annotation data is required for training [325].

While we tested a variety of different classifiers, such as Random Forests (RFs), K-Nearest Neighbors (KNNs) and even a single-layer perceptron, we found that Support Vector Machines (SVMs) worked best for the prediction tasks we were interested in. This makes sense, considering that SVMs are well-suited for datasets with a large number of features and a rather small number of samples, as long as the training data quality is good enough [325]. Since we only train on Uniprot and GO data that has been manually verified by expert curators which leads to datasets with fewer than 50 samples per class, as well as feature vectors with upwards of 1,000 positions before feature selection, SVMs are our primary choice as a predictive model.

2.2.3.1 Support Vector Machines

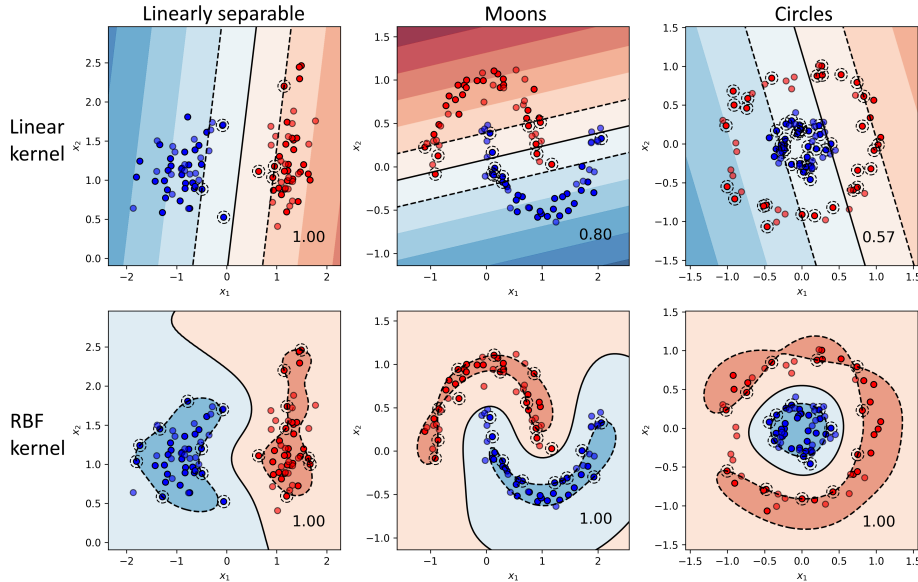


Figure 2.19: Decision function, decision hyperplane (straight line) and margins (dashed lines) of a linear SVM ($C = 1$) and a RBF-kernel SVM ($C = 1, \gamma = 1$) trained and evaluated on three datasets, containing $n = 100$ samples (50 per class) and $d = 2$ features each. Every change in contour level of the background color denotes an increase or decrease of the decision function result by one; it is equal to zero at the decision boundary. Negative values of the decision function, as well as the negative class, are colored in red. The three datasets were each split into a training- and test set, in a stratified fashion. Both the training and test sets were standardized using a z-score function derived from the mean and standard deviation of training set. The SVMs were then optimized on the training data, and evaluated on the test set. Samples from the test set have a lighter color, and samples from the darker-colored training set that act as support vectors are marked by dashed circles. The mean test accuracy, i.e. the average percentage of correctly classified samples across the equally-sized classes of the test set, is written in the bottom right of each plot. Decision function values should only be used as a confidence estimate between samples in the same plot, as the scales are not normalized between the six examples. Dataset creation, SVM training and visualization were performed with *scikit-learn* [56] and *matplotlib* [52].

Given n samples (e.g. n proteins) that are encoded by n *feature vectors* of length d (such as fixed-size encodings that represent the protein sequence or 3D structure), as well as a binary class label for each sample (e.g. “sugar transporter” and “amino acid transporter”), a Support Vector Machine (SVM) [340] aims to find a hyperplane (meaning a $(d - 1)$ -dimensional object in d -dimensional space, like a straight line in 2D space, or a flat plane in 3D) that separates the n samples in space, ideally corresponding to the class labels (see top-left panel in Figure 2.19) [325].

More formally, the SVM is provided with a sample/feature matrix $\mathbf{X} \in \mathbb{R}^{n,d}$ and a vector $\mathbf{y} \in \{-1, 1\}^n$ of known class labels, where -1 and 1 denote the negative and positive classes, respectively¹. Through a training process, the SVM finds

¹Whereas the underlying math typically uses class labels $y \in \{-1, 1\}^n$ [341][340][342], other literature [325], as well as some software packages [56], use the class labels 0 and 1 instead, as this is the general convention for binary classifiers.

a hyperplane defined by the equation

$$\mathbf{w}^T \mathbf{x} + b = 0$$

, where $\mathbf{w} \in \mathbf{R}^d$ is a vector of *weights*, $b \in \mathbb{R}$ is the *bias* that determines the offset of the hyperplane from the origin, and $\mathbf{x} \in \mathbb{R}^d$ is any feature vector from sample space². During the training process, fixed values for $\mathbf{w}$ and b are determined based on the features and class labels in the training dataset $(\mathbf{X}, \mathbf{y})$. Subsequently, the *decision function* f for a feature vector $\mathbf{x}_i \in \mathbb{R}^d, i \in \{1, \dots, n\}$ is given as:

$$f(\mathbf{x}_i) = \mathbf{w}^T \mathbf{x}_i + b$$

The absolute value $|f(\mathbf{x}_i)|$ of this function can be interpreted as a relative confidence score for the prediction, since it is proportional to the shortest distance between the d -dimensional point $\mathbf{x}_i$ and the decision-making hyperplane at $f(x) = 0$. Dividing this absolute value by $\|\mathbf{w}\|$ then gives us the Euclidean distance (this is important later). After determining $\mathbf{w}$ and b from the training data, a *predicted* class label $\hat{y}_i \in \{-1, 1\}$ for a sample $\mathbf{x}_i$ can now be derived from the decision function:

$$\hat{y}_i = \begin{cases} -1 & \text{if } f(\mathbf{x}_i) < 0, \\ 1 & \text{if } f(\mathbf{x}_i) \geq 0 \end{cases}$$

The vector $\hat{\mathbf{y}} \in \{-1, 1\}^n$ contains the predictions for all n samples in $\mathbf{X}$ [325].

Around the central hyperplane $\mathbf{w}^T \mathbf{x} + b = 0$ that carries out the classification task, the SVM defines two parallel *boundary hyperplanes*

$$\begin{aligned} \mathbf{w}^T \mathbf{x} + b &= 1 \\ \mathbf{w}^T \mathbf{x} + b &= -1 \end{aligned}$$

, which together create a *margin* (see the three lines in the top-left panel of Figure 2.19). In the original *hard-margin* approach to training SVMs, the underlying optimization problem is solved in such a way that the samples with the shortest distance to the central hyperplane, the name-giving *support vectors*, are located exactly on these boundary hyperplanes, and no samples are allowed to be inside the margin. Since this method is very sensitive to outliers, and also makes it impossible to train on datasets that are not linearly separable (such as the top-middle plot in Figure 2.19), Cortes and Vapnik [340] developed the concept of a *soft margin*, which allows a number of samples to be inside the margin, and can cause support vectors to not be located exactly on the margin boundary. This soft margin is controlled through the regularization³ parameter C , where lower values allow for more samples to enter the margin around the central hyperplane. Samples are penalized for entering the margin through a *loss function*, e.g. *hinge loss* or *squared hinge loss*. The loss penalizes samples for entering the margin, while the C parameter controls how much the SVM optimization algorithm is influenced by those penalties [342]. Lowering C can therefore reduce overfitting on the training data, by limiting

²Some software packages that provide access to these methods, such as scikit-learn [56], commonly refer to $\mathbf{w}$ and b as *coefficients* and *intercept*, respectively.

³The process of generalizing a model by making it less sensitive to individual samples, in turn preventing overfitting on the training data [325].

the influence of individual samples on the width of the margin and position of the decision boundary [325]. As the loss function is calculated as a sum over all samples (discussed later in this section), a larger class contributes more to its result than a smaller class, effectively pushing the hyperplane towards the smaller class (e.g. when the full dataset has 50 positive and 500 negative samples). To counteract this problem, it is common to multiply the parameter C with a different *class weight* for each class. Samples from the smaller class get a larger weight and vice versa, typically it is inversely proportional to the class size [342][56].

While the soft-margin approach theoretically makes training possible on any feature dataset, there are still many cases where a simple hyperplane does not yield a usable predictive model, as the classes are not linearly separable (e.g. second and third panel on the first row of Figure 2.19). One solution to this problem is to map the feature dataset to a higher-dimensional space, where a hyperplane is then capable of linearly separating the classes. For example, a feature vector with a single feature dimension x_1 could be mapped to a higher-dimensional space by adding a second feature $x_2 = \phi(x_1) = x_1^2$ through a function $\phi(x)$. The resulting parabola in $\mathbb{R}^2$ can now be cut into two pieces by a straight line (which is the shape of a hyperplane in $\mathbb{R}^2$), allowing more datasets to be linearly separated. This is called a (second-degree) polynomial feature transformation [325].

The actual SVM training process solves an optimization problem that tries to maximize the width of the margin by finding appropriate values for the aforementioned weight vector $\mathbf{w}$ and bias b , while also including the regularization parameter C , the class weights and the loss function, that together potentially allow a portion of samples to enter the margin (see e.g. the top-center panel in Figure 2.19)[342]. After putting everything so far together, we arrive at the primary optimization problem for a SVM with soft margin, feature transformation, class weights and l_2 norm penalty [342]:

$$\begin{aligned} \min_{\mathbf{w}, b, \xi} \quad & \frac{1}{2} \|\mathbf{w}\|^2 + C^+ \sum_{y_i=1} \xi_i + C^- \sum_{y_i=-1} \xi_i \\ \text{subject to} \quad & y_i(\mathbf{w}^T \phi(\mathbf{x}_i) + b) \geq 1 - \xi_i \\ & \xi_i \geq 0, \quad i \in \{1, \dots, n\} \end{aligned}$$

, where $\mathbf{w} \in \mathbb{R}^d$ is the vector of weights (coefficients), and $b \in \mathbb{R}$ is the bias (intercept), both of which get optimized. Minimizing the *penalty term* $\frac{1}{2} \|\mathbf{w}\|^2$ under the constraint effectively maximizes the width of the margin [325]; This would be the only objective in the hard-margin problem. As mentioned before, the Euclidean distance between any sample $\mathbf{x}_i \in \mathbb{R}^d$ and the central hyperplane is calculated as

$$\frac{|\mathbf{w}^T \mathbf{x}_i + b|}{\|\mathbf{w}\|}$$

. Minimizing the penalty term also minimizes $\|\mathbf{w}\|$, thereby maximizing the distance⁴. The remaining *loss* terms in the objective function then introduce the soft margin, where $C^+, C^- \in \mathbb{R}$ are the regularization parameters for the positive and negative class, after applying the two class weights to the parameter C . The symbol $y_i \in \{-1, 1\}$ denotes the class (positive or negative) of sample $i \in \{1, \dots, n\}$, and $\xi_i \in \mathbb{R}$ is its result of the loss function that penalizes samples inside the margin. The regularization parameter C therefore adjusts how much influence the loss function has on the objective. The feature vector of sample i is $\mathbf{x}_i \in \mathbb{R}^d$, and ϕ is a feature transformation function, such as the polynomial feature transformation mentioned above⁵. Finally, the constraint effectively forces samples to stay outside the margin, as the result of the decision function

$$f(\mathbf{x}_i) = \mathbf{w}^T \phi(\mathbf{x}_i) + b$$

for a sample located *inside* the margin would be in the range $(-1, 1)$, due to the fact that the margin hyperplanes are defined by $f(\mathbf{x}) = \pm 1$. Including ξ_i in the constraint potentially allows a number of samples to break that rule, leading to a soft margin. A common loss function is the *hinge loss*:

$$\xi_i = \max(0, 1 - y_i \cdot f(\mathbf{x}_i))$$

, which increases linearly after a sample $\mathbf{x}_i$ with class $y_i \in \{-1, 1\}$ enters the margin space defined by $|f(\mathbf{x}_i)| < 1$, and is 0 when the sample is located on or outside the margin boundary. As the name suggests, the alternative *squared hinge loss* function is calculated as the square of the hinge loss [343]. Hinge loss and squared hinge loss are also commonly referred to as L1-loss and L2-loss, respectively, while using the Euclidean l_2 norm in the penalty term is called *L2-regularization*, and replacing it with the Manhattan norm l_1 would be L1-regularization [343].

Unfortunately, the approach of directly transforming the features with the function ϕ does not scale very well, and becomes very computationally intensive for larger datasets. There is, however, a way to achieve the same, or an even more optimal result, through a mathematical trick. To this end, the SVM optimization problem described above is replaced by its *dual problem*, which is a different constrained optimization problem that, in case of SVMs, has the same optimal solution as the original *primal problem*⁶ of finding $\mathbf{w}$ and b . The dual problem can be directly derived from the primal problem [341][325]. One big advantage of this replacement is that *kernel functions* can now be introduced into the optimization objective. This *kernel trick*, which was introduced by Boser et al. [341], allows us to create a SVM that behaves as if the samples had been feature-transformed into a higher-dimensional space where they are (potentially) linearly separable, without actually having to do that, which considerably reduces the computational complexity [325]. Alternatively, *kernel approximation* methods [344], such as Nyström [345] and Random Fourier

⁴We minimize $\frac{1}{2}\|\mathbf{w}\|^2$ term instead of just $\|\mathbf{w}\|$ because that expression would not be differentiable at $\mathbf{w} = 0$, whereas the former simply has $\mathbf{w}$ as its derivative. Optimization algorithms work better on differentiable functions [342][325]. The penalty is also commonly written as the equivalent $\frac{1}{2}\mathbf{w}^T \mathbf{w}$ [342]

⁵The feature transformation could transform $\mathbf{x}_i$ to a higher-dimensional space $\mathbb{R}^{d'}$, $d' > d$, in which case that the weight coefficients $\mathbf{w}$ would also be in $\mathbb{R}^{d'}$

⁶The primal problem needs to meet a set of conditions to have a dual problem with the same optimal solution, which that of the SVM optimization problem does [325]

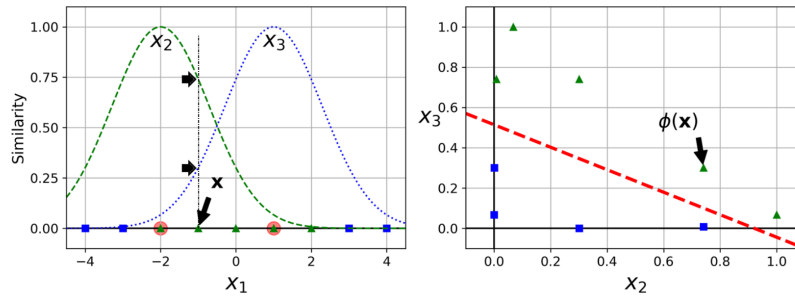


Figure 2.20: Figure adapted from Géron [325] that shows an example for a feature transformation with a radial basis function (RBF). In the left plot, nine samples belonging to two classes (blue squares and green triangles) only have one numerical feature x_1 , with which they can not be linearly separated. Two samples (red circles) are selected as landmarks, and bell curves are drawn with those samples as their mean, while the variance is inversely proportional to the RBF parameter γ . For each remaining sample, the similarity is calculated according to the RBF function, and these similarity scores now make up the new, higher-dimensional feature space in the right plot, which is linearly separable by a 2D hyperplane (i.e. a straight line).

Features (RFF) [346], can be used together with linear, primal-problem SVMs as described above, where they approximate the transformation matrix of a kernel based on training data, and fully transform the data before passing it on to the linear SVM. This delivers faster results and scales much better with larger datasets, but can lead to different results, and introduces an additional parameter [345][346].

Besides a linear kernel that behaves as the linear SVM described above, and a polynomial kernel that corresponds to the polynomial feature transformation, another commonly found kernel relies on a radial basis function (RBF). The idea behind this approach is to select a subset of samples as *landmarks*, and then measure the distance between any sample $\mathbf{x}$ and each landmark $\mathbf{l}$ according to the function

$$\phi_\gamma(\mathbf{x}, \mathbf{l}) = (-\gamma \|\mathbf{x} - \mathbf{l}\|^2)$$

, where $\|\mathbf{x} - \mathbf{l}\|^2$ is the squared Euclidean distance between feature vectors $\mathbf{x}$ and $\mathbf{l}$ (see Figure 2.19) [342]. With $\gamma = \frac{1}{2\sigma^2}$, this is essentially a Gaussian function (i.e. a bell curve) with mean value $\mathbf{l}$ and variance σ^2 (see Figure 2.20). Larger values of γ therefore create a bell curve around $\mathbf{l}$ that is more narrow [325]. The result of this similarity measure is a value in $[0, 1]$, where a value of 0 is reached when the other feature vector $\mathbf{x}$ is located outside of the bell curve defined by γ and $\mathbf{l}$, and $\phi_\gamma(\mathbf{x}, \mathbf{l}) = 1$ means that $\mathbf{x}$ is located directly in the middle ($\mathbf{x} = \mathbf{l}$). By limiting the width of the bell curve around each landmark, a larger γ can limit the influence that each sample has on the newly created feature dimensions and therefore on the decision boundary, meaning that γ is another regularization parameter that can reduce overfitting, similarly to C [325]. How exactly C and γ work together to cause or prevent overfitting is shown in Figure A.5, where we sampled and visualized the decision functions of different parameter combinations on a toy dataset that is not linearly separable in its original form, using a SVM with RBF kernel.

Many datasets have more than two labels, meaning that the samples are not just assigned to a positive and negative class, but to three or more. We might, for example, be interested in not just telling apart sugar transporters from amino acid transporters, but also include a third or fourth substrate class that a protein can carry across a membrane. Formally, this would mean there are no more positive and negative class labels $y \in \{-1, 1\}$, but instead each of the m classes is assigned a label $y \in \{0, \dots, m-1\}$. While SVMs are not inherently multi-class, they do support this type of classification through different types of voting schemes. In the one-vs-rest (OvR) approach, a binary SVM classifier is trained to distinguish one class from all others, and this process is repeated for all classes. The final prediction is typically selected based on average normalized confidence scores. With the one-vs-one (OvO) method, a SVM is trained for each combination of two classes, and the class that was selected the most in these pairwise comparisons is voted as the overall result. While the latter method usually leads to more optimal results, as the “rest” class in OvR can be quite noisy, it also comes with an increased computational complexity, as the number of SVMs that need to be trained grows exponentially with the number of classes in the dataset ($\frac{m(m-1)}{2}$ models for m classes). However, the runtime of SVMs also grows exponentially with the size of the training dataset, it can be in $O(n \times d)$ for highly optimized linear SVM algorithms [343], and up to $O(n^3 \times d)$ for kernelized SVMs with RBF kernel [342], when training on a dataset of n samples and d features. Therefore, a OvR approach where the “rest” class is very large can often be slower [325].

In the first half of this chapter, we found that many transport proteins carry two or more substrates, and in many cases, this is reflected by the amino acid residues in their binding pockets. This means one protein can be in multiple classes at the same time. One solution to that problem is *multi-label* classification. In practice, this means that the m class labels $y \in \{0, \dots, m-1\}$ from the multi-class problem are now replaced by n binary vectors $\mathbf{y} \in \{-1, 1\}^m$, which are arranged into a target matrix $\mathbf{Y} \in \mathbb{R}^{n \times m}$. The element $\mathbf{Y}_{i,j}$ is then a binary value that denotes whether sample i is a member of class j . Some machine learning algorithms inherently support multi-label classification, such as a deep neural network (DNN) that has multiple nodes in its output layer. Multi-output classification is a type of multi-label classification where all classes in $\mathbf{y}$ are treated independently (which is typically not the case for multi-label DNNs), meaning that the model can not learn from co-occurrences of classes. We can create a SVM-based multi-output classifier by training a binary SVM for each of the m classes in the dataset. Another option for implementing a multi-label SVM would be to calculate class probabilities during training, through *Platt scaling* [347], which has been extended for the multi-class case [348]. Each position in the target vector is then assigned to the positive or negative class according to its probability score (e.g. by applying a threshold, or a window around the highest score) [325][56].

Since distances are calculated based on real numbers from the feature vectors, SVM training can be influenced by the scales of the individual features. The feature vector positions are therefore brought to the same scale, most commonly by replacing them with their z-scores, or by linearly scaling them to the same range, e.g. to $[0, 1]$ (the latter of which is more susceptible to outliers) [325].

2.2.3.2 Feature selection and dimensionality reduction

One issue with biological datasets is that they often have many features, but few samples available for training. Each reliable sample first requires costly lab-experiments and subsequent peer-reviewed publications. Inferred or predicted data can increase the sample size, but potentially decrease reliability. On the other hand, as soon as we have a protein sequence and know its function, we can encode this sequence with various algorithms [349], find sequence domains or families [300][350], find or predict the 3D structure with the PDB [264] or AlphaFold [274] and then create structural features [351], and much more.

While SVMs excel at these types of datasets compared to other classification algorithms in terms of runtime complexity (see Section 2.2.3.1), reducing the number of feature dimensions can still be beneficial to both the training time and prediction accuracy. Noisy and/or redundant data, where the number of feature dimensions d is much greater than the number of samples n , can cause the model to fit to the noise, instead of finding patterns that have statistical relevance for the prediction. This so-called *curse of dimensionality* can cause models to over-fit⁷ on their training data, as well as increases the usage of computational resources during training and prediction [325].

Methods that reduce the number d of feature dimensions of each sample can be classified into two different categories. On one hand, *feature selection* removes the same positions from each feature vector according to a metric, while keeping the rest of the original vector. On the other hand, methods that fall under *dimensionality reduction* map the original d -dimensional sample space to a new coordinate system with a lower number of dimensions, replacing the original vector with a new one. In both cases, the goal is to keep information that is relevant to the predictive model, while removing redundancy and noise. While decreasing the number of elements in the feature vectors can be one solution to the curse of dimensionality, others would be to increase the number of samples through oversampling methods from e.g. the *imbalanced-learn* package [352], or to identify outliers through hierarchical clustering [353].

Possibly the most commonly used dimensionality reduction technique in machine learning is the *principal component analysis* (PCA) [354]. The first principal component, often labeled PC1, is the axis along which the highest variance can be found in the dataset. The second principal component is then perpendicular to the first, and points towards the largest amount of remaining variance. In total, PCA finds d principal components in a d -dimensional dataset, where PC_i is perpendicular to the first $i-1$ principal components $PC1, \dots, PC_{i-1}$. Now we can calculate the amount of variance in the data that is explained by the first $i \leq d$ principal components as a running sum, starting with PC1, and then stop adding more PC's when enough variance is explained by $PC1, \dots, PC_i$. If $i < d$ and the explained variance is close to 100% (e.g. more than 95%) then we have successfully reduced the number of dimensions, without losing much information in the process [325]. In practice, the number of principal components to choose is often determined through hyperparameter optimization, as explained

⁷Overfitting on the training data means that the model only works well on this specific dataset, whereas a *generalized* model can also be applied to previously unseen data.

in Section 2.2.3.5. The first two or three principal components are often used to create plots of higher-dimensional datasets and classifiers [325].

One method for *feature selection* already comes with the *liblinear* [343] SVM implementation, in the form of L1-regularization [355]. In the SVM optimization problem from Section 2.2.3.1, the expression $\frac{1}{2}\|\mathbf{w}\|^2$ is the penalty term that aims to maximize the margin. As per convention, the l_2 -norm $\|\mathbf{w}\|_2$, which is the Euclidean distance between $\mathbf{w}$ and the origin, is often written as $\|\mathbf{w}\|$. Generally, an l_p -norm is defined as

$$\|\mathbf{w}\|_p = \left(\sum_{i=1}^d |w_i|^p \right)^{\frac{1}{p}}$$

in the d -dimensional sample space. Replacing the (Euclidean) l_2 norm in the objective function with the (Manhattan) l_1 norm (i.e. replacing $\frac{1}{2}\|\mathbf{w}\|^2$ with simply $\|\mathbf{w}\|_1$ [355]) encourages sparsity in the weight coefficient vector $\mathbf{w}$. In a sparse weight vector, multiple positions are set to zero, meaning the corresponding positions in a feature vector $\mathbf{x}$ are multiplied with those zeroes in the decision function $f(\mathbf{x}) = \mathbf{w}^T \mathbf{x} + b$, and no longer affect the result. As this mostly happens to features that are redundant, the 1-norm SVM effectively comes with built-in feature selection [355]. The current standard SVM implementations [342][343][56] do not support L1 regularization and the kernel trick at the same time, meaning that kernel *approximation* methods [344] such as Nyström [345] or Random Fourier Features [346] would have to be used to transform datasets that are not linearly separable to a higher-dimensional space, before passing them on to a linear SVM. This can speed up the training time, but can also deliver different results than kernelized SVM models that calculate the genuine kernel matrix [345][346].

Univariate feature selection involves the calculation of a score for each individual feature, such as the ANOVA F-score (Section 2.2.2.4). Each score is a measure for how well the feature can separate the classes, and the k highest ranking features, or the top $p\%$, are selected for the model, while the remaining feature vector positions are removed from all samples. To achieve meaningful evaluation results, the positions have to be selected based on training data, without knowledge of which values these features have in the subset used for testing. As mentioned before, the null hypothesis of ANOVA F-test is that the mean values of a feature $\mathbf{x}_i$ are the same across all target classes that the samples belong to, whereas the alternative hypothesis is that it is different between *any* pair of these sample groups. To this end, we calculate the F-score, which is the variance of the feature *between* groups, divided by the variance of the feature *within* groups. Larger scores can therefore be used as evidence against the null hypothesis [334]. Instead of calculating p-values, one typically optimizes the parameters k or p through hyperparameter optimization, which is explained in Section 2.2.3.5. An alternative to the F-score for non-negative and whole-numbered features is the χ^2 -score, which is the test statistic of the non-parametric χ^2 test [356].

2.2.3.3 Evaluation

The purpose of evaluating a supervised machine learning model is to estimate how well it fits the given dataset, and simulate its performance on previously unknown data. To create the SVM models for Figures 2.19 and A.5, we split the dataset into two parts: one part for training, and one part for testing the trained model. It is important that the model does not know anything about the test set before the evaluation, and models should ideally not be modified to optimize the score on the test set. To avoid information-sharing with the testing data, any constants used by pre-processing methods, such as mean and standard deviation for the z-score, should be derived from the training set, and then applied in the same way to both subsets. For smaller datasets, a *stratified* split can help in getting more accurate evaluation results, by making sure that the ratio between classes in the subsets remains as close as possible to that in the full dataset [325].

After performing the test, we can now compare the predicted classes for the test set to the known classes, to see how many samples were classified correctly. Each sample is assigned to one of four groups: the true and false positives (TP and FP), as well as the true negatives and false negatives (TN and FN). A sample in the test set that actually belongs to the positive class is either a TP or a FN, depending on whether it was classified correctly, whereas a negative sample is either a true negative or a false positive.

A wide selection of metrics exist that are calculated from these four counts, and the best choice of metric depends on the dataset at hand. Possibly the simplest metric is the *accuracy* [325], which is equivalent to the percentage of correctly classified samples:

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

One problem with this metric is that it does not work very well for imbalanced datasets. If 95% of the dataset belongs to the positive class, then a classifier that *always* classifies any sample as positive would have a 95% accuracy.

Precision and recall are often calculated together, as they give a good overview of the model's predictive performance [357]:

$$\text{Precision} = \frac{TP}{TP + FP} \quad \text{Recall} = \frac{TP}{TP + FN}$$

The precision score describes the reliability of positive *predictions*, whereas a higher recall means that the model has good classification performance on *actual* positive samples in the dataset (which are either classified as TP or FN). There is a trade-off between precision and recall, where an increase of one of them leads to a decrease of the other [325]. In the example of a medical diagnosis test for a disease, the precision tells us how much we can trust a positive diagnosis, the recall or sensitivity denotes the likelihood that a sick patient is diagnosed correctly as positive.

To get a balanced model that sits between those two options, we can optimize for the F_1 score, which is defined as the harmonic mean between precision and

recall [357]:

$$F_1 = \frac{2 \cdot \text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} = \frac{2TP}{2TP + FP + FN}$$

Using the harmonic mean makes sure that both precision and recall have to be high for F_1 to have high values [325].

Since none of these three scores pay attention to the true negatives in any way, in those cases where that is important, they are often used in conjunction with the *specificity*:

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

, which describes the performance of the model on negative samples from the test set [357]. In the previous example of the medical diagnosis test, a higher specificity means that healthy (actual negative) patients are more likely to be classified as healthy (predicted to be negative).

Recall is also called *sensitivity* [357], and the arithmetic mean of sensitivity and specificity is the *balanced accuracy* score [358]:

$$\text{Balanced Accuracy} = \frac{(\text{Sensitivity} + \text{Specificity})}{2}$$

When the two classes can not be seen as a “true vs. false” test result, for example in cases where we try to predict if a transporter protein carries substrate **A** or **B**, then we need to calculate some of these scores in either direction, and take the average. This is necessary because F_1 , for example, does not treat positives and negatives in the same way, meaning that always labeling **A** as the positive class and **B** as negative, or vice-versa, would lead to misleading results. There are three different methods for calculating the average F_1 score: The macro-average is calculated for each class individually:

$$F_1^{\text{macro}} = \frac{1}{m} \sum_{i=1}^m F_1^i$$

, while the *micro*-average is calculated from all of the TP/TN/FP/FN across all classes instead. A middle-ground between the two is the weighted macro-average, which calculates F_1 scores for each class individually, but then scales them according to the number of elements in that class, meaning that smaller classes have smaller contribution to the score. The macro-average is preferred for imbalanced datasets where the class size is not relevant, as it treats all classes equally [325]. For our substrate prediction models, we therefore calculate the macro-averaged F_1 .

Whereas the balanced accuracy ignores the false positives, and the F_1 score does not pay attention to false negatives, there is a score that combines all of them into one. The Matthews Correlation Coefficient (MCC) [359] returns a value in $[-1, 1]$, and can be interpreted as the correlation between the predictions $\hat{\mathbf{y}}$ and the known true values of the test set $\mathbf{y}$:

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

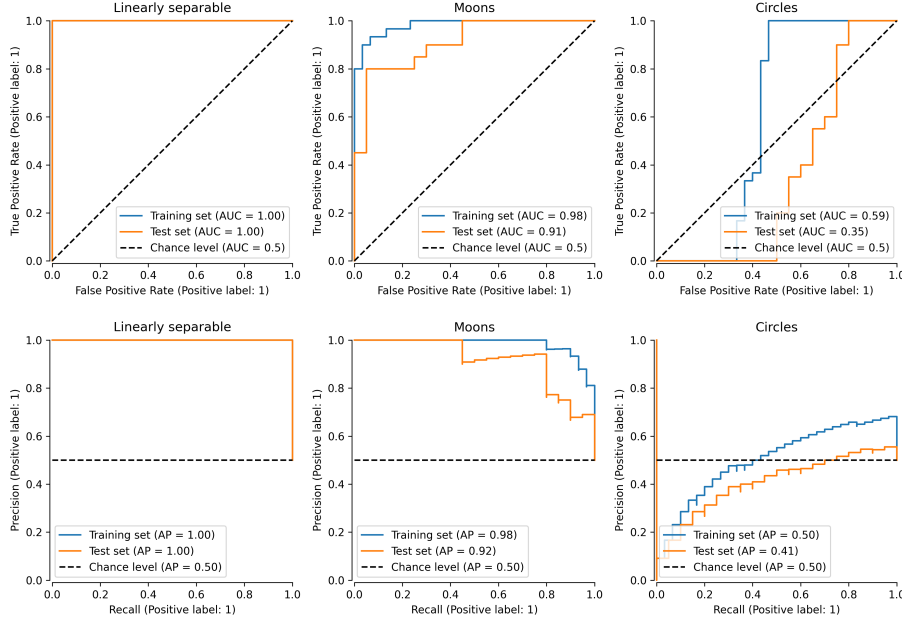


Figure 2.21: ROC and PR curves for the linear SVM model from Figure 2.19. The corresponding metrics were calculated from the decision function, with class label 1 as the positive and 0 for the negative class. The linearly separable dataset shows an example of a perfect elbow curve, the classifier on the moons dataset still shows adequate performance, and the SVM on the circles dataset is close to a dummy classifier that performs a uniform random draw (dotted line).

The MCC is equivalent to the Pearson correlation coefficient (see Section 2.2.2.6) between two binary variables [357]. The previously mentioned metrics return a value of 0.5 for a classifier that makes predictions based on a uniform random distribution in cases where the dataset is perfectly balanced (i.e. all classes are equally large), and a lower value simply means that predictions are *worse* than the random classifier. In contrast to that, the MCC can actually find a negative correlation between the predictions and the actual labels (negative values), while the uniform random classifier will lead to no correlation (close to zero), and a positive correlation has a positive MCC.

A common way to visualize the trade-off between precision and recall is to draw a precision-recall (PR) curve (see bottom row in Figure 2.21). Instead of using the actual class labels $\mathbf{y}$ and the predicted labels $\hat{\mathbf{y}}$ to count the TP/TN/FP/FN and derive the score from that, the PR curve is calculated from the true values $y_i, i \in 1, \dots, n$, and their decision function values $f(\mathbf{x}_i)$, where n is the number of samples, and $\mathbf{x}_i$ is the feature vector for sample i . Sweeping through all observed values of the decision function then allows us to visualize how well the selected decision boundary can separate the classes. The PR plot can be converted into a performance metric $\text{PR-AUC} \in [0, 1]$, by calculating the area under the curve. The resulting score is also called *average precision* (AP).

Replacing the precision in the PR plot with the *false positive rate* (FPR) ($\frac{FP}{FP+TN}$) turns it into the more commonly used receiver-operator characteristic

(ROC) curve (see top row in Figure 2.21). In this case, the recall is referred to as the true positive rate (TPR), which is a third name for the score besides sensitivity. This curve can also be converted into a performance metric in the range $[0, 1]$ by calculating the area under the curve (ROC-AUC). The PR curve is usually preferred in cases where the positive class contains few samples, or when false positives are more important than false negatives. In the former case, the ROC curve can be inaccurate [325]. Since some of our substrate classes have a very low sample count, optimizing models with the macro-averaged F_1^{macro} score would be most appropriate.

In conclusion, the one-sided precision, recall and F_1 scores are a good choice when caring more about the positive class than about the informative value of negative predictions, since the true negatives (and the total number of samples) are ignored during their calculation. This is especially useful when the positive class only has a small number of samples available, as those will have a larger influence on the results compared to other metrics. Calculating their macro-average then makes those scores applicable to multi-class applications, as well as to binary tasks between two classes that represent categories (e.g. sugar vs. amino acid transporter), instead of positive or negative test results. Since the specificity of the positive class is equal to the recall of the negative class, the F_1^{macro} score also includes the true negatives, albeit indirectly. The balanced accuracy shows the average performance of the model on the actual samples, e.g. the arithmetic mean of how likely a sick patient is classified as sick, and a healthy patient as healthy. The MCC allows us to analyze the linear relationship between actual values and the predictions for the test set. In contrast to the other scores, we can also find a strong negative correlation. On the other hand, it is not possible to take this score apart into two separate scores, such as sensitivity/specificity or precision/recall, or to compare the one-sided F_1 /recall/precision to their respective macro-averages in order to gain additional insights. Finally, the ROC-AUC and AP scores (see Figure 2.21) give us a measure for how well the optimized decision boundary can separate the two classes, at different thresholds of the decision function.

2.2.3.4 Cross validation

Regardless of the performance metric used to evaluate the classifier, the evaluation results can still depend on the choice of the training set. For example, there might be a sub-group of samples in a class that form their own cluster, and not including enough of these samples in the training set can put them on the wrong side of the decision boundary when predicting classes for the test set. To get an estimate for how well a model generalizes on a dataset, and potentially uncover problems with the dataset creation and pre-processing methods, we can perform multiple of these train-test splits, and then calculate the average score across all of them. Splitting the dataset k times, in a way where each individual sample appears in the test set exactly once across all splits, is called a k -fold cross validation. In a *stratified* cross validation, the fraction of positive and negative samples is kept as close as possible to that of the original dataset. To get an even better estimate of generalization, the dataset can be shuffled multiple times, a cross validation is then applied after each shuffle, and an average score across all shuffles is calculated. If $k = n$ splits are performed in a dataset

of n samples, meaning that each test set only contains one sample, then that is called a leave-one-out cross validation (LOOCV), which can be especially useful for datasets with few samples [325].

In order to achieve a reliable evaluation, some parameters need to be determined on the training set only, to avoid information from the test set influencing the model optimization process. This includes, for example, the mean and average for the z-score with which each feature is standardized, the number k of best features to keep according to the ANOVA F-score, or the number of principal components selected. Pipelines are data structures that allow for those steps, such as preprocessing, feature selection, oversampling and SVM classification, to be combined into one *model*, which can fit and transform training and test data separately [56][325].

2.2.3.5 Hyperparameter optimization

Over the last sections, the machine learning model has accumulated several parameters, including the type of pre-processing method to apply (e.g. standardization or scaling to a range), the number k of best features to keep according to the ANOVA F-score (or how many principal components to take), the cost parameter C that controls the SVM soft margin, and the gamma parameter of RBF kernel that determines the influence of each individual sample on the decision boundary. As shown in Figure A.5, optimizing these parameters can be quite important for a good model performance, and for prevention of overfitting and underfitting. One option would be to reduce the number of dimensions to the first two or three principal components and then create a number of plots to compare. A more reliable and automated approach for higher-dimensional data is to try all combinations of parameters. In *grid search*, a parameter grid is provided, containing one or multiple values for each model parameter (e.g. in Figure A.5, that grid was $C \in \{0.1, 1, 10\}$, $\gamma \in \{0.1, 1, 10, 100\}$). The model then tries all possible combinations of parameters and evaluates the resulting models using a performance metric. To avoid overfitting, a k -fold cross validation is performed, and scores for all parameter combinations are calculated for each individual train/test split. The best set of parameters is then finally selected based on the highest average score across all k cross-validation splits [56][325]. The performance metric with which the model is evaluated can be selected to achieve a particular goal, e.g. F_1^{macro} or balanced accuracy to get a model that performs equally well on all classes, or recall to eliminate false negatives in a medical diagnosis test (see Section 2.2.3.3).

Chapter 3

Optimized Data Set and Feature Construction for Substrate Prediction of Membrane Transporters

The contents of this chapter were published in the Journal of Chemical Information and Modeling, under the title *Optimized Data Set and Feature Construction for Substrate Prediction of Membrane Transporters* [349]. The raw data, along with all the code that was implemented, is available for download at <https://github.com/adenger/subpred3>.

3.1 Abstract

Alpha-helical transmembrane proteins termed membrane transporters mediate the passage of small hydrophilic substrate molecules across biological lipid bilayer membranes. Annotating the specific substrates of the dozens to hundreds of individual transporters of an organism is an important task. In the past, machine learning classifiers have been successfully trained on pan-organism datasets to predict putative substrates of transporters. Here, we critically examine the selection of an optimal data set of protein sequence features for the classification task.

We focus on membrane transporters of the three model organisms *E. coli*, *A. thaliana*, *S. cerevisiae*, as well as human. We show that organism-specific classifiers can be robustly trained if at least 20 samples are available for each substrate class. If information from position-specific scoring matrices is included, such classifiers have F1 scores between 0.85 and 1.00. For the largest dataset (*A. thaliana*), a 4-class classifier yielded an F-score of 0.97. On a pan-organism dataset composed of transporters of all four organisms, amino acid and sugar transporters were predicted with an F1 score of 0.91.

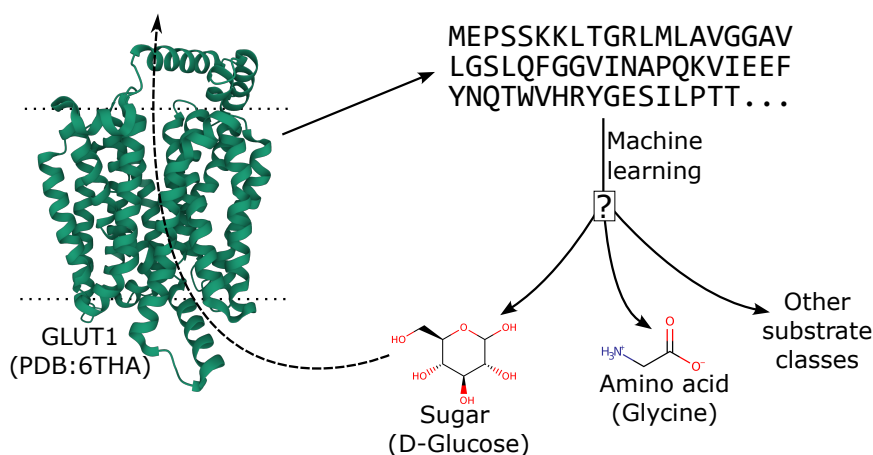


Figure 3.1: Graphical abstract outlining the general idea behind the project. After determining the amino acid sequence of a transport protein, it is encoded into a machine learning (ML) feature. A supervised model has been trained on sequences of proteins, along with their known substrates. The trained algorithm can then predict the substrate class of the protein from its sequence, e.g. to tell whether it carries sugars or amino acids.

3.2 Introduction

Membrane transporters are integral membrane proteins that mediate the passage of molecules across biological membranes. In case of substrate uptake, they typically enable the passage of one or a few types of ions, amino acids, lipids, or sugars into cells. Exporters may have broader specificities, such as xenobiotics. Identifying the specific substrates of a given transporter by the means of experimental assays is difficult and labor intensive. Hence, bioinformatics analysis may be helpful in predicting putative substrates, which can be tested subsequently in targeted experiments.

An important data source in this field is the Transporter Classification Database (TCDB) [118], which provides structural, functional, mechanistic, evolutionary and disease/medical information for more than 10,000 non-redundant transport systems, and categorizes them into over 1,000 transporter families. TransportDB [297] provides precompiled genome-wide information about annotated membrane transporters for more than 2,500 organisms, including their substrates. The UniProt database [360] provides transporter and substrate annotations as well in their *UniProt Keywords* database, and through Gene Ontology (GO) [47] annotations.

In the past years, machine learning (ML) methods have been successfully applied in the field of protein functional annotation [361]. ML methods were found to be capable of predicting the transport mechanisms of membrane transporters [362], and their putative substrates [363], with good success.

By analyzing transport proteins categorized according to the TCDB system, Barghash and Helms [38] found that sequence similarity is suitable for predicting broad categories, such as the transport mechanism, but less so for determining

individual substrates. The latter is more likely to rely on small recognition sites, which can influence the frequency of some amino acid types in the sequence, through their residue composition [364].

So far, the prediction of transported substrates was mostly carried out using sequence-derived ML features. While motifs associated with membrane proteins show a relatively high conservation, substrate specificity is often determined by a small subset of sequence positions and amino acid types [365, 26]. Ideally, this difference in sequence composition between the substrate classes (i.e. the transport protein sets corresponding to the transport of a particular substrate) is then recognized by the machine learning algorithm.

Schaadt and Helms [366][39] generated protein fingerprints based on amino acid composition (AAC) and pair amino acid composition (PAAC) of transporters in *A. thaliana*. Redundant sequences were removed if their sequence identity exceeded 90%. The proteins were classified into four substrate classes (amino acids, oligopeptides, phosphates, and hexoses), with an accuracy of about 90%.

Then, using a pan-organism dataset derived from UniProt and applying a 70% sequence similarity threshold, Mishra et al. [40] trained a Support Vector Machine (SVM) that classified transport proteins into seven substrate classes (amino acid, anion, cation, electron, protein/mRNA, sugar, and other) based on AAC, PAAC, physico-chemical composition, biochemical composition, and position-specific scoring matrices (PSSM), with an accuracy between 67% and 84%. Subsequently, Alballa et al. [367] used the same dataset and achieved accuracy scores between 89 and 97%, as well as a Matthews Correlation Coefficient (MCC) of 0.69.

As mentioned above, most previous authors have analyzed pan-organism datasets of transporter-substrate annotations. However the influence of individual organisms, the issue of sequence homology between training and test data set, and the effect of outliers were not systematically assessed.

In this work, we revisit the task of classifying membrane transporters according to substrate classes based on a dataset compiled from a recent release of UniProtKB/Swiss-Prot. We focused on four comparatively well-annotated organisms, namely *E. coli*, *S. cerevisiae*, *A. thaliana* and human. We tested how well the organism-specific classifiers work in comparison to a multi-organism classifier, and examined the influence of sequence homology on the results. Also, we present a data pipeline for creating specialized membrane transporter datasets from UniProt, and our strategies for removing outliers.

The main findings of our study are that the sample count in training data sets matters strongly, and that organism-specific models are not well transferable to other organisms. Both arguments speak in favor of training substrate prediction models on multi-organism data sets, for the time being. This will be particularly helpful when turning to deep learning methods in the future.

3.3 Methods

The developed pipeline for membrane transporter substrate prediction consists of three steps. First, a dataset of transmembrane transport proteins is com-

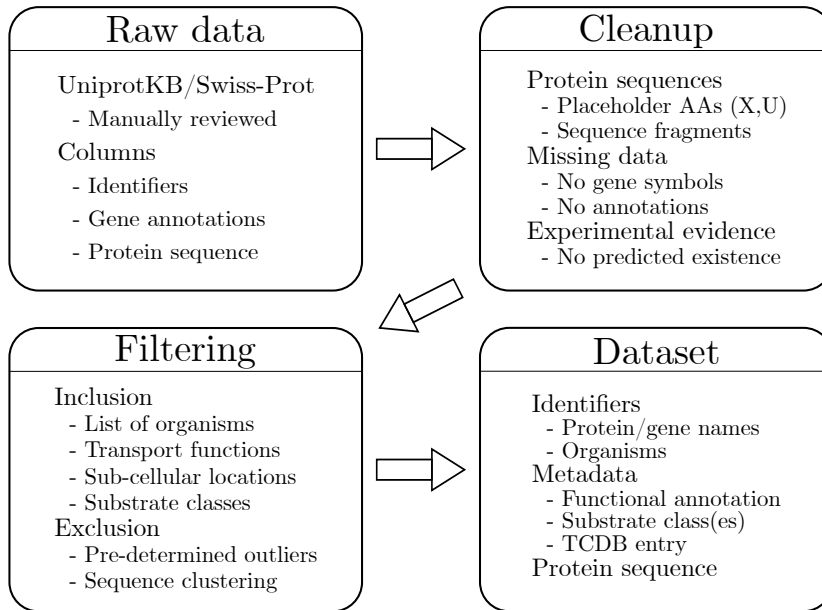


Figure 3.2: The membrane transporter dataset creation pipeline. A custom dataset is downloaded from UniProtKB/Swiss-Prot, with the necessary columns. After removing samples with missing data, and excluding proteins without experimentally proven existence, the dataset is filtered with several inclusion- and exclusion-lists. The final dataset contains identifiers, gene/protein annotation data, cross-references to other relevant databases, as well as the protein sequence.

piled from publicly available data sources, and filtered according to the user’s specification. The biological properties of these proteins are then encoded into fixed-size feature vectors. Finally, a machine learning model is trained on the data, and tested in a cross-validation approach, using the *sklearn* Python package [56].

This model can then be used to predict substrate classes of previously unknown transport proteins, from the sequence alone or by incorporating additional data if it is available.

Jupyter notebooks are provided for all three steps (automatic compilation of dataset, computation of features, training of the classifiers).

3.3.1 Dataset creation

The dataset of membrane transport proteins with known substrates is derived from the manually curated Swiss-Prot database [360]. First, all proteins whose existence has not been experimentally proven at transcript or protein level, as well as proteins with incomplete sequences, are removed from the dataset. Next, the dataset is filtered for proteins that are annotated with a set of UniProt keywords. These keywords belong to three categories: 31 *substrate keywords* (e.g. *sugar transport*, *amino acid transport*), 16 *transport-related keywords* (e.g. *antiport*, *ion channel*), and 11 *location keywords*, such as *cell membrane* or *mitochondrion*. The full list of keywords is given in the supplementary

material as Appendix B.2. The dataset can also be filtered for proteins from a specific organism, or a list of organisms, by providing a list of taxonomy identifiers. The dataset is then annotated with data from related databases, such as the Transporter Classification Database (TCDB) [118], and the Gene Ontology (GO) [47]. An overview of the dataset generation pipeline is provided in Figure 3.2

Finally, the dataset is clustered based on sequence identity to reduce redundancy on the sample axis, using the program *cd-hit* [317].

3.3.1.1 Handling of multilabel data

Some transport proteins are annotated with multiple substrates. While most previous studies, such as those of Mishra et al. [40] and Alballa et al. [367], decided to remove all proteins with more than one reported substrate from the training dataset, we implemented two additional ways of dealing with multilabel proteins. Which of these methods is the most suitable one depends on the dataset at hand.

The first option would be to keep the multi-label proteins, and create new classes for them. This is only useful when enough samples are available for the new combined class, e.g. sodium/potassium pumps.

The second option, called *integrate*, only removes multi-substrate transporters from the dataset if more than one of these substrates is the subject of the classification task (e.g. turning amino acid/ion symporters into amino acid transporters, when only trying to tell amino acid transporters apart from sugar transporters). This can increase the number of available data points, but has the potential to add outliers to the dataset.

A third option would be the usage of a classifier that supports multi-label data, such as K-nearest-neighbors (KNN). Due to the low sample availability, this approach is not feasible for single-organism datasets. It might be interesting for transporter datasets with a larger number of proteins per class, e.g. in multi-organisms models.

3.3.2 Machine learning features

3.3.2.1 Amino acid compositions

The amino acid composition (AAC) represents the frequency of each natural amino acid in a given protein sequence, normalized by the length of the protein sequence [39][368].

The dipeptide composition [368], also called PAAC short for pair amino acid composition, counts the occurrences of each dipeptide formed from natural amino acids, i.e. AA, AC, AD, ... , YV, YW, YY. The result is divided by the total amount of possible dipeptides in the protein, which corresponds to the length of the sequence minus one. Due to the nature of peptide bonds, the dipeptides a-b and b-a are not considered as equivalent, if a and b are two different amino acids.

Since there are 20 natural amino acids, the AAC is made up of 20 elements, while the PAAC feature vector has $20 \times 20 = 400$ positions.

3.3.2.2 PSSM feature

A position-specific scoring matrix (PSSM) [369] of a protein contains 20 scores for each position in its amino acid sequence.

In order to create a feature vector with constant length from a PSSM, the rows for each amino acid type are added together along the entire protein sequence, divided by the length of the sequence, and scaled to the interval [0,1]. This method is commonly used to prepare PSSMs for machine learning tasks [370].

PSSM calculation is carried out locally with the *psiblast* executable from the BLAST+ package [299, 316]. Local computation allows for parallel processing and caching, which drastically speeds up the computation time. Two BLAST databases are considered, created from the Uniref50 and the Uniref90 databases with the *makeblastdb* executable. We specified an E-value threshold of 0.002 for inclusion of proteins in the PSSM profile, and generated PSSMs with different numbers of psiblast iterations, in order to find more closely or more distantly related proteins.

The optimal number of psiblast iterations and choice of BLAST database (Uniref50, Uniref90) are selected automatically, based on their classification performance on the respective training set. The algorithm also combines multiple PSSM features into one, if that improves the model fit on the training data. The final feature vector has 400, 800, 1200 or 1600 dimensions, depending on the combination of PSSMs that was selected during the optimization step.

3.3.3 Machine learning pipeline

After generating a dataset of membrane transporters with the pipeline described in Section 3.3.1, and encoding the proteins as machine learning features using the feature generation algorithms described in Section 3.3.2, a supervised learning model is trained and evaluated on the data. All tasks related to machine learning are carried out with the *scikit-learn* Python package [56].

The *substrate keywords* that were specified during the dataset creation process are used as class labels.

In all cases, a Support Vector Machine (SVM) is used for classification. If necessary, the redundancy and noise on the feature axis was reduced through a Principal Component Analysis (PCA). Alternatively, a feature selection algorithm based on ANOVA F-values was applied.

For evaluation, the dataset is split into a training set and an independent test set, in a stratified fashion. The parameters of the feature generation algorithms and machine learning algorithms are optimized on the training set, by finding the set of parameters with the highest average score during a cross validation. Finally, the pipeline is trained on the whole training dataset, and the performance of the model is tested on the independent test set.

The model is optimized with the macro-averaged F1 score, in order to achieve an optimal average predictive performance for all individual classes. The precision, recall and F1 scores for the individual substrate classes are used for evaluating the classification.

The whole process is repeated 20 times, with different random seeds. The standard deviation between these tests gives an estimate of the dependence on the first train-test-split.

For datasets with fewer than 20 samples in any of the classes, the model is also evaluated with a leave-one-out cross validation (LOOCV), in order to maximize the number of samples in the training set.

Multiclass classification is carried out through a one-vs-one voting scheme.

As supplemental files, we provide Jupyter notebooks that deterministically carry out the generation of datasets, computation of features and machine learning tasks described in this study. The code can be found on Github, under <https://github.com/adenger/subpred3>

3.4 Results

3.4.1 Distinguishing amino acid transporters from sugar transporters in *A.thaliana*

3.4.1.1 Dataset creation

The transporter dataset was created with the pipeline described in Section 3.3.1. Proteins were filtered for taxonomy identifier 3705 (*A. thaliana*), location keyword *Membrane*, function keyword *Transport*, and substrate keywords *Amino acid transport* and *Sugar transport*.

3.4.1.2 Dataset analysis: Outliers

After creating the dataset, we calculated the amino acid composition for all proteins, in order to analyze the feature dataset.

Since AAC, PAAC and PSSM are all based on the amino acid sequence of a protein, the analysis of the simpler AAC should also translate to the PAAC and PSSM features, in terms of classification performance.

When visualizing the Pearson correlation matrix of all proteins in the dataset as a heatmap with hierarchical clustering, it became clear that seven amino acid transporters in the dataset had a very low correlation to the remaining transporters (see Figure 3.3a).

We also performed a PCA on the AAC dataset, and found that the same proteins, which belong to the *Glutamine Dumper* (GDU) family, are assigned a much higher value on the first principal component than the rest of the dataset (see Figure 3.3b).

Additionally, a Shapiro Wilk test on the individual amino acid frequencies showed that two of these frequencies, those for Pro and for Glu, did not follow a normal distribution. This was caused by an unusually high Pro and Glu content in those seven proteins, compared to other amino acid transporters in *A. thaliana* (see Figure 3.3c, d). All of them belong to the GDU family.

GDU1 only contains a single transmembrane helix [371], GDU2-7 are paralogs of GDU1 [372]. According to the 2D and 3D structures predicted by Pilot et al.

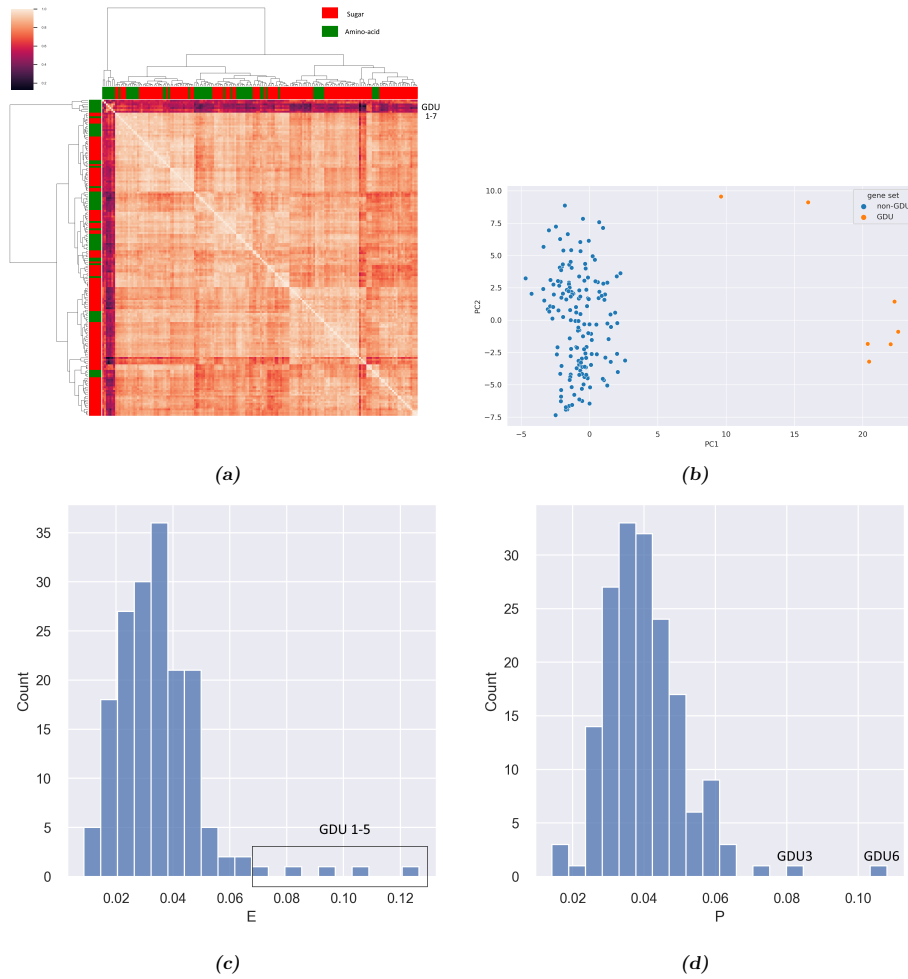


Figure 3.3: (a) Heatmap of *A. thaliana* AAC correlation matrix, with hierarchical Ward-clustering, before removal of outliers. (b) PCA of the AAC dataset shows how the GDU proteins are separate from the rest of the dataset (c, d) The GDU proteins are responsible for the fact that Pro and Glu do not follow a normal distribution, such as the rest of the dataset.

[371] and AlphaFold [21][274], respectively, the seven GDU proteins have a very similar structure. While GDU expression correlates with Gln efflux, it is still unknown whether the GDU proteins are part of a membrane transport complex, or if they facilitate transport via other exporters [373].

The TCDB assumes the former, and classifies GDU1 as a channel protein (TCDB class 1). On the other hand, all remaining proteins in the dataset were either classified as TCDB class 2.A (porters driven by electrochemical gradient), or their amino acid composition had a Pearson correlation of at least 0.8 to a transporter in class 2.A. The correlation to the GDU proteins was only around 0.2-0.4. Due to their different structure (i.e. single-pass membrane protein), their different amino acid composition and their unknown transport mechanism, we considered the GDU proteins as outliers and did not include GDU1-7 in our machine learning model.

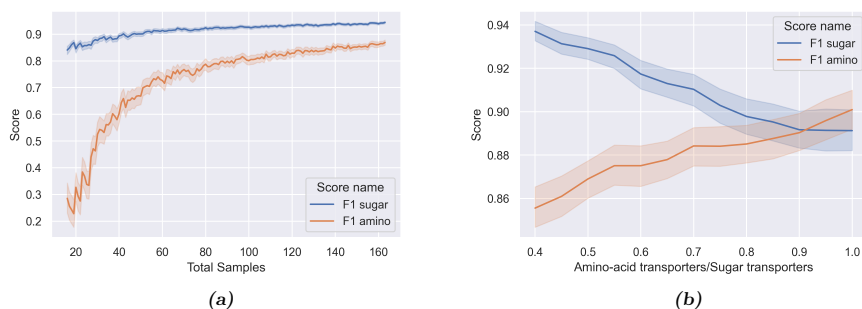


Figure 3.4: (a) *A. thaliana* AAC Dataset was downsampled in a stratified fashion, to keep the sample ratio between the classes. Downsampling was repeated 50 times with multiple random seeds. An RBF-SVM was used for classification. (b) Only the majority class in the *A. thaliana* AAC dataset was randomly down-sampled. Downsampling was repeated 50 times with different random seeds. Classification was done with an RBF-SVM. The x-axis is the ratio between minority class (amino acid transporters) and majority class (sugar transporters)

Other outliers were not found.

3.4.1.3 Effect of sample count on classification performance

In order to estimate the minimum number of samples necessary to create a meaningful machine learning model, we evaluated a Support Vector Machine classifier with a radial basis function kernel (RBF-SVM) on the *A. thaliana* AAC-dataset with a 5-fold cross validation, while lowering the number of samples in the dataset from 165 to 10 in steps of 1.

First, the number of samples was lowered in a stratified fashion, in order to keep the same number of samples per class as in the complete dataset. The random sampling was repeated with 50 different random seeds. Figure 3.4a shows the mean F1 scores for both classes. The observed trend suggests that at least a total 80 samples in the dataset, corresponding to 57 sugar transporters and 23 amino acid transporters, is necessary for training a reliable machine learning model on our dataset.

Since amino acid transporters consistently received lower F1 scores than sugar transporters, while also being represented by fewer samples in the dataset, we tested if lowering only the number of sugar transporters through downsampling would have any effect on this difference. Therefore, the percentage of sugar transporters in the dataset was lowered in 13 steps, from 71.5% down to 50%. A RBF-SVM was evaluated on the amino acid transporters and the down-sampled sugar transporters, through a 5-fold cross validation (see Figure 3.4b). The random downsampling followed by a cross validation was repeated 50 times with different random seeds. Decreasing the number of sugar transporters in the dataset also decreased the F1 score for predicting sugar transporters. This suggests that the higher F1 score for sugar transporters compared to amino acid transporters in Fig. 3.4a was related to the fact that more sugar transporters were available for training.

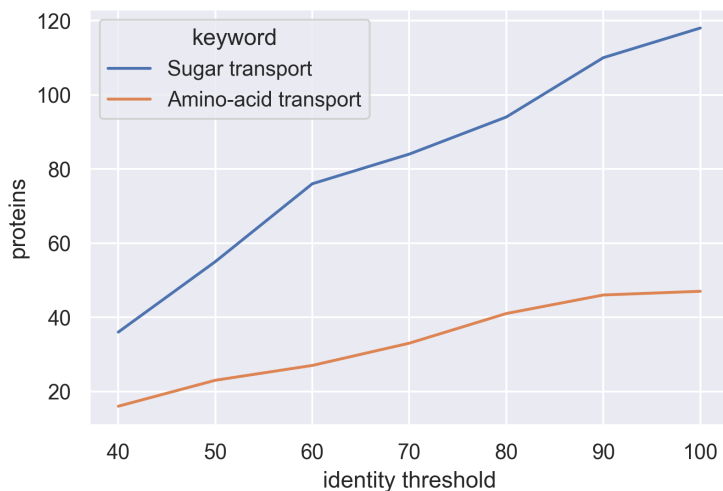


Figure 3.5: Number of transport proteins in each class when applying *cd-hit* to the dataset of sugar transporters and amino acid transporters in *A. thaliana* at different sequence identity thresholds.

3.4.1.4 Effect of sequence clustering on classification performance, sample count and redundancy

Sequence clustering was carried out with *cd-hit* at six different thresholds, ranging from 40% to 100%.

While there were 47 amino acid transporters and 118 sugar transporters in the dataset before clustering, only 16 and 36 were left at a 40% sequence identity threshold, respectively (see Figure 3.5).

Since the suitability of the dataset for membrane transporter substrate prediction decreases considerably when the number of samples in the minority class is below 30 (see Figure 3.4a), we chose an identity threshold of 70% for all remaining analyses, unless specified otherwise. This coincides with the threshold used by Mishra et al. [40], who also created a dataset of membrane transporters that was based on Swiss-Prot.

3.4.1.5 Amino-acid composition (AAC)

To test the classification performance of the AAC feature, we trained a Support Vector Machine (SVM) with a RBF kernel. The feature data was standardized, and the number of dimensions was reduced through Principal Component Analysis (PCA) to those principal components that can explain 98% of the variance in the dataset. Constants for standardization and PCA were derived from the training set, and applied to the test set. The model achieved a macro-averaged F1 score of 0.89 in the training set, and 0.88 on the independent test set.

We analyzed the importance of each amino acid type for the classification task by calculating the ANOVA F-scores between a given amino acid type and the class labels (i.e. the transported substrates). The most important feature for distin-

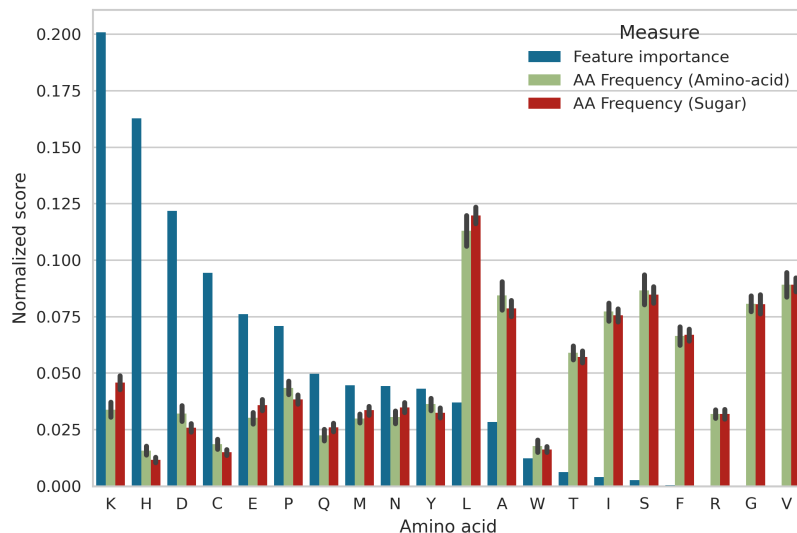


Figure 3.6: Feature importances for each amino acid type when distinguishing sugar transporters from amino acid transporters in *A. thaliana* using the AAC feature, compared to amino acid frequencies in the transport protein datasets. Feature importance scores were normalized to the same scale by dividing them by their sum.

guishing between sugar transporters and amino acid transporters in *A. thaliana* was the frequency of lysine, followed by histidine, aspartic acid and cysteine. Lysine occurs more frequently in sugar transporters, whereas histidine, aspartic acid and cysteine are found more often in amino acid transport proteins (see Figure 3.6). Feature importance shows a negative Pearson correlation of -0.452 to the AAC, suggesting that substrate specificity is mostly determined by less frequently occurring amino acids in these sequences. The most frequent amino acid types in the dataset are hydrophobic and have a high alpha-helical propensity, which makes sense considering that all proteins considered here contain multiple transmembrane alpha-helix domains. The five amino acid types most commonly found in electrochemical potential-driven transporters according to Gromiha and Yabuki [362] (F, G, I, L, V) were among the seven highest amino acid frequencies in our dataset.

3.4.1.6 Paired amino-acid composition (PAAC)

With the PAAC feature, we trained a machine learning pipeline comprised of standardization, PCA, and a RBF-SVM classifier. It achieved a macro-averaged F1 scores of 0.936 on the training set, and 1.0 on the independent test set.

The three dipeptides with the highest feature importance were cysteine-thyrosine (CY), phenylalanine-leucine (FL) and glycine-aspartic acid (GD). Asp and Cys were among the four most important AAC features (see Figure 3.6).

This suggests that the PAAC feature encodes additional information that is

not captured by the AAC. While leucine and phenylalanine do not show high predictive performance on their own, they are useful for distinguishing sugar transporters from amino acid transporters when occurring as the dipeptide Phe-Leu.

3.4.1.7 Position-specific scoring matrix (PSSM)

To predict substrates based on their position-specific scoring matrices, we calculated four PSSMs for each protein, where we used either one of the two BLAST-databases available (Uniref50 and Uniref90), and either one or three iterations of PSI-BLAST. The best combination of PSSM parameters was determined through automatic hyperparameter optimization with a custom transformer, in a 5-fold cross validation on the training set. A RBF-SVM, together with standardization and feature selection based on F-Scores, was used for training and evaluation. The pipeline classified all transporters correctly, both in the cross validation on the training set and when evaluated on the independent test set.

Out of the 50 feature vector positions with the highest feature importance according to the ANOVA F-score between the training set and test set, 26 positions corresponded to exchange rates of the amino acid histidine. Since the individual vector elements can be interpreted as the likelihood of one amino acid type being exchanged for another throughout evolution, this observation suggests that evolutionary histidine conservation is an important measure for distinguishing amino acid transporters from sugar transporters. In our dataset, both the histidine content and histidine conservation are higher among amino-acid transporters, compared to sugar transporters. These findings match our analysis of the AAC feature above, where histidine was the amino acid type with the second-highest feature importance. After histidine, the second and third most common amino acid types in the top 50 PSSM feature positions were cysteine (10) and phenylalanine (8), which are also among the most important PAAC features (see above).

In order to estimate the choice of the independent test set on the results, we repeated the analysis 20 times, with 20 different random seeds for splitting the dataset. The average macro-averaged F1 scores were 0.997 and 0.968 for the cross validation on the training set and for the independent test set, respectively. The corresponding standard deviations were 0.0057 for the training set, and 0.0449 for the independent test set.

3.4.1.8 Meta-feature based on sequence features

Finally, we combined all previous models into one meta-feature. A RBF-SVM was trained on the AAC, PAAC and PSSM features. Standardization and PCA were performed on the training set, and applied to the test set.

The model achieved a macro-averaged F1 score of 0.987 during the cross validation, and 1.00 on the independent test set.

Again, we estimated the influence of independent test set selection on the F1 scores. The average F1 scores were 0.993 on the training set, and 0.954 on the test. The standard deviations across the 20 repetitions were slightly higher than for PSSM, with 0.0071 and 0.045 for the training set and test set, respectively.

Among the 50 features with the highest feature importance, there were 43 PSSM features, 6 PAAC features, and 1 AAC feature. This suggests that all feature types contribute to the meta-feature in some way. However, the classification performance was slightly lower than that of the PSSM feature on its own, at least when distinguishing sugar transporters from amino acid transporters in *A. thaliana*.

3.4.2 Application to other organisms

Next, we applied the methods described in Section 3.4.1 to three additional organisms: *E. coli*, *S. cerevisiae*, and *H. sapiens*. This was done to test the approach on a wider variety of species, i.e. multi-cell and single-cell eukaryotic organisms, as well as a gram-negative bacterium, with an inner and an outer cellular membrane.

3.4.2.1 Differences in subcellular locations

In case of eukaryotes, which also have transporters localized to the membranes of various cellular compartments such as the endoplasmatic reticulum (ER) and mitochondria (MT), the inclusion of ER and MT proteins yielded an increased classification performance, most likely due to the increased sample count, and did not introduce any outliers. Hierarchical clustering analysis of the transporter amino acid composition showed that transporters of the membranes of MT, ER and other intracellular membranes do not stand out in terms of their amino acid composition, when compared to proteins in the cellular membrane, at least for the sugar- and amino acid transporters that we analyzed. In contrast, the protein/mRNA transporters in the nuclear membrane, also called *Nuclear Pore Complexes (NPC)*, have a distinct method of transport that is not shared by cellular membrane protein transporters, and hence, were not included in the datasets. MT membranes in eukaryotes are also known to contain beta-barrel proteins [374], although that was not the case in our datasets.

For *E. coli*, a gram-negative bacterium, there were only five transporters in our sugar/amino acid transporter dataset that were located in the outer membrane. All of those proteins were pores that transport sugars and ions. In terms of their amino acid composition, these proteins were clear outliers, with a Pearson correlation coefficient of less than 0.4 to the rest of the dataset. In contrast, the AACs of the remaining transporters had pairwise correlation coefficients of more than 0.8. This makes sense, considering that general transporter motifs such as hydrophobic alpha helices often show high conservation, whereas substrate specificity is determined by a small subset of sequence positions [365, 364].

3.4.2.2 Outliers

Compared to the seven outliers in the *A. thaliana* dataset, only two were found for human, and none for yeast. The first outlier in *H. sapiens* was **rBAT**, which, according to the TCDB, is an *accessory factor to transport* having one transmembrane domain and no known substrates. The other one was **SLC38A10**, a protein that contains a strongly conserved amino acid transport domain that encodes 10 transmembrane helices and is 398 amino acids in length, but also roughly 700 additional amino acids with unknown structure or function, which

influences its AAC. In *E. coli*, we found the five aforementioned pores in the outer membrane, as well as 16 peripheral membrane proteins, which can be filtered out by using the *Transmembrane* keyword instead of *Membrane*. These peripheral membrane proteins are typically associated with transmembrane transport complexes, e.g. *ATP binding cassettes* (ABCs), but do not facilitate the actual transport themselves. Including their sequences in the dataset would introduce noise and could lead to worse classification performance of the sequence-based features.

3.4.2.3 Multi-substrate transporters

Some transporters in the datasets transport more than one substrate according to their annotation, e.g. symporters and antiporters. Mishra et al. [40] decided to remove all such proteins with more than one known substrate. Our pipeline also has an option to remove all labels from the dataset which are not used as labels for classification, see Section 3.3.1.1. As previously mentioned, this did not make a difference in the sugar/amino acid transporter dataset from *A. thaliana*, since it did not contain any multi-substrate transport proteins for these two substrates. The same was the case for *S. cerevisiae*.

The *integrate* method represents a trade-off: Having more samples in the dataset can increase classification performance if the new samples match the rest of the dataset, but can also introduce noise if they do not. In case of the *E.coli* dataset, for example, the integrate method removed the keyword *Ion transport* from the dataset of sugar and amino acid transporters. *E.coli* contains five outer-membrane porins that transport small sugars and ions, and those were grouped together with sugar uniporters. The porins were clear outliers in the dataset, since outer-membrane porins are made up of beta barrel structures, whereas passive transporters typically contain multiple transmembrane helices. On the other hand, the method found two additional amino-acid transporters (putP, gltS) and one sugar transporter (uhpT), all of which are passive transporters located in the inner membrane. These three additional proteins had amino acid compositions similar to those of other transporters in the dataset, and increased the sample count for training and testing. For *H. sapiens*, we found 21 additional samples for our training dataset. Most of these proteins were sodium-dependent symporters (14 samples), followed by four chloride-dependent amino acid exporters, and four sodium-coupled amino acid transporters that transport neurotransmitters.

3.4.2.4 Comparison to TCDB

Our approach of creating the transporter dataset from Swiss-Prot data increased the number of available samples significantly, compared to cross-referencing TCDB with Swiss-Prot. The numbers of additional transmembrane transporters in the dataset without TCDB entry were 108, 4, 3 and 9 for *A. thaliana*, *E. coli*, *H. sapiens* and *S. cerevisiae*, respectively.

3.4.2.5 Effect of sequence clustering with cd-hit

Sequence clustering was applied to all four transporter datasets. Again, there was a trade-off between the number of samples that are available for training,

Organism	No clustering			70% clustering		
	Sugar	Amino acid	Σ	Sugar	Amino acid	Σ
<i>A. thaliana</i>	118	47	165	84	33	117
<i>E. coli</i>	48	51	99	47	51	98
<i>S. cerevisiae</i>	28	36	64	17	34	51
<i>H. sapiens</i>	36	51	87	34	49	83

Table 3.1: The effect of sequence clustering with cd-hit on sample sizes for sugar and amino acid transporters in the four organisms.

and the clustering threshold. As shown before in Figure 3.5, the classification performance of the machine learning model drops off sharply once the number of transporters in a particular class falls below 20-30. Therefore, we chose a sequence identity threshold of 70% for all organisms, which is the lowest value that still yields adequate evaluation results.

Interestingly, the number of proteins removed by clustering varies widely between organisms. In *E. coli*, only one transporter was removed. The highest number of redundant transporters was found in *A. thaliana* (see Table 3.1). According to its documentation, cd-hit always chooses the protein with the longest sequence as the cluster representative.

Clustering was carried out after removing the outliers.

3.4.2.6 Choice of machine learning algorithms and preprocessing methods

When comparing the classification performance between different algorithms, i.e. SVMs with different kernels, KNN, Random Forests, Naive Bayes and stochastic gradient descent, the only algorithm that consistently delivered good results was the Support Vector Machine. In most cases, the RBF kernel delivered the best results. However, in some cases we decided to use a linear kernel instead, in order to reduce overfitting on the training data. The fact that SVMs perform the best makes sense, considering that they usually work well on datasets with many features and few samples, as long as the number of outliers is not too high.

In many cases, performing a principal component analysis (PCA) in order to reduce the number of feature dimensions improved the classification results. Typically, only a small percentage of positions in the protein sequence are actually responsible for substrate specificity (see Figure 3.6). Machine learning features calculated from the entire protein sequence can therefore contain a lot of useless and redundant information. This is especially true for the meta-feature, which is made up of multiple fingerprints of the same protein sequence. When PCA was applied, the feature dimensions were reduced to a number of principal components that explain somewhere between 80% and 99% of the variance in

Organism	AAC		PAAC		PSSM		META	
	train	test	train	test	train	test	train	test
<i>A. thaliana</i>	0.888	0.877	0.936	1.000	1.000	1.000	1.000	1.000
<i>E. coli</i>	0.813	0.850	0.792	0.792	0.908	0.899	0.907	0.950
<i>H. sapiens</i>	0.659	0.757	0.822	0.757	0.936	1.000	0.903	0.938
<i>S. cerevisiae</i>	0.769	0.670	0.913	0.771	0.879	0.612	0.913	0.612

Table 3.2: Macro-averaged F1 scores for all features and organisms. The train score is the average F1 score achieved during cross validation. The test score was calculated by applying the trained model on an independent test set.

Organism	Model	AAC		PAAC		PSSM		META	
		train	test	train	test	train	test	train	test
<i>H. sapiens</i>	SVM	0.712	0.623	0.756	0.693	0.924	0.859	0.901	0.859
	PCA+SVM	0.749	0.627	0.882	0.839	0.957	0.884	0.955	0.857
<i>S. cerevisiae</i>	SVM	0.838	0.723	0.802	0.728	0.880	0.843	0.862	0.794
	PCA+SVM	0.845	0.655	0.919	0.880	0.951	0.818	0.929	0.754

Table 3.3: Additional evaluation for organisms having few labeled transporters. Two models were trained: a support vector machine (SVM), as well as a pipeline of principal component analysis (PCA) followed by SVM classification. Evaluation was carried out through nested leave-one-out cross validation (LOOCV), meaning that one transporter was used as an independent test set, and the remaining transporters for hyperparameter optimization through another LOOCV. The scores shown here are the average F1 scores from all iterations of the outer cross validation (*test*), as well as the average training scores achieved during the inner cross validations (*train*).

the data. The constants for the PCA were derived from the training data, and applied to the test data.

Since both SVMs and PCA typically perform better when the data is standardized, we included this preprocessing step in every pipeline.

3.4.2.7 Evaluation results

The evaluation results are shown in Table 3.2. During the evaluation, some of the organisms reached a relatively low number of samples, due to the fact that the independent test set already made up 20% of the data, and a 5-fold cross validation was carried out on the remaining 80%.

For *S. cerevisiae*, only 17 sugar transporters were left in the dataset after clustering, which means that only 10 ($0.8 \times 0.8 \times 17$) transporters were available for training during the cross validation.

Test set	Feature	Amino			Sugar		
		Correct	Incorrect	Accuracy	Correct	Incorrect	Acc.
<i>A. thaliana</i>	AAC	33	0	1.00	82	2	0.98
	PAAC	33	0	1.00	84	0	1.00
	PSSM	33	0	1.00	84	0	1.00
<i>H. sapiens</i>	AAC	20	28	0.42	29	2	0.94
	PAAC	4	44	0.08	34	0	1.00
	PSSM	21	27	0.44	31	3	0.91
<i>E. coli</i>	AAC	14	37	0.27	45	2	0.96
	PAAC	1	50	0.02	45	2	0.96
	PSSM	8	43	0.16	47	0	1.00
<i>S. cerevisiae</i>	AAC	10	24	0.29	14	3	0.82
	PAAC	8	26	0.24	17	0	1.00
	PSSM	9	25	0.26	17	0	1.00

Table 3.4: Classification results when training a machine learning model on transporters from *A. thaliana*, and applying that model to other organisms. We conclude that sequence-based models trained on *A. thaliana* are not transferable to other organisms.

Therefore, we carried out a second evaluation, this time using nested leave-one-out cross validation, in order to maximize the number of training samples. In most cases, this led to improved scores and less overfitting on the training set (see Table 3.3).

3.4.3 Datasets involving more than one organism

3.4.3.1 Application of model trained on one organism to another

Due to the low availability of samples for some of the organisms, we considered the possibility of training a model on an organism with a comparatively larger number of samples, and then using that model to predict substrates in an organism with fewer samples. We chose the sugar/amino acid prediction task again, along with a machine learning pipeline consisting of standardization and a SVM with RBF kernel.

As a first test, we trained the model only on transporter data from *A. thaliana*, and then predicted substrates in all four organisms (see Table 3.4). The model had a good fit on the training data from *A. thaliana*, with F1 scores of 0.98 or higher. When applied to the datasets from other organisms, most sugar transporters were classified correctly ($accuracy \geq 0.82$), but most amino-acid transporters incorrectly ($accuracy \leq 0.44$). We conclude that the SVM hy-

Organism		Amino			Sugar		
Training	Test	Correct	Incorrect	Acc.	Correct	Incorrect	Acc.
<i>E. coli</i>	<i>E. coli</i>	51	0	1.00	44	3	0.94
	<i>H. sapiens</i>	28	20	0.58	8	26	0.24
	<i>S. cerevisiae</i>	24	10	0.71	9	8	0.53
	<i>A. thaliana</i>	28	5	0.85	28	56	0.33
<i>H. sapiens</i>	<i>E. coli</i>	17	34	0.33	36	11	0.77
	<i>H. sapiens</i>	45	3	0.94	32	2	0.94
	<i>S. cerevisiae</i>	15	19	0.44	9	8	0.53
	<i>A. thaliana</i>	25	8	0.76	58	26	0.69
<i>S. cerevisiae</i>	<i>E. coli</i>	51	0	1.00	0	47	0.00
	<i>H. sapiens</i>	42	6	0.88	5	29	0.15
	<i>S. cerevisiae</i>	34	0	1.00	17	0	1.00
	<i>A. thaliana</i>	31	2	0.94	23	61	0.27

Table 3.5: Classification results when training a machine learning model on transporters from organisms other than *A. thaliana*, and applying that model to all four organisms. Sequence-based features for distinguishing sugar transporters from amino-acid transporters in one organism can not be applied to another organism. Accuracy was shortened to “Acc.”.

perplane trained on *A. thaliana* feature data (AAC, PAAC and PSSM) is not transferrable to other organisms. In order to rule out the possibility of overfitting on the training set, we repeated the test with a linear SVM. However, the scores did not improve.

In a second test, we trained a model on each of the other three organisms (see Table 3.5). Since the scores between the three sequence-based features seem to correlate, we only performed this test using the AAC, in order to reduce the number of individual test cases. The classification performance did not improve when using organisms other than *A. thaliana* for training.

3.4.3.2 Meta-organism models

After studying the individual datasets for the four organisms, we decided to also combine their transporters into one meta-organism.

An overview of this dataset is provided in the form of a heatmap of amino acid compositions, along with hierarchical Ward-clustering (see Figure 3.7). *E. coli* and *S. cerevisiae* both had their own, distinctive sub-clusters, whereas *H. sapiens* and *A. thaliana* were more similar to each other. We calculated pairwise correlations to find outliers in the data. The outliers in the meta-organism were the same proteins that we previously removed from the individual datasets:

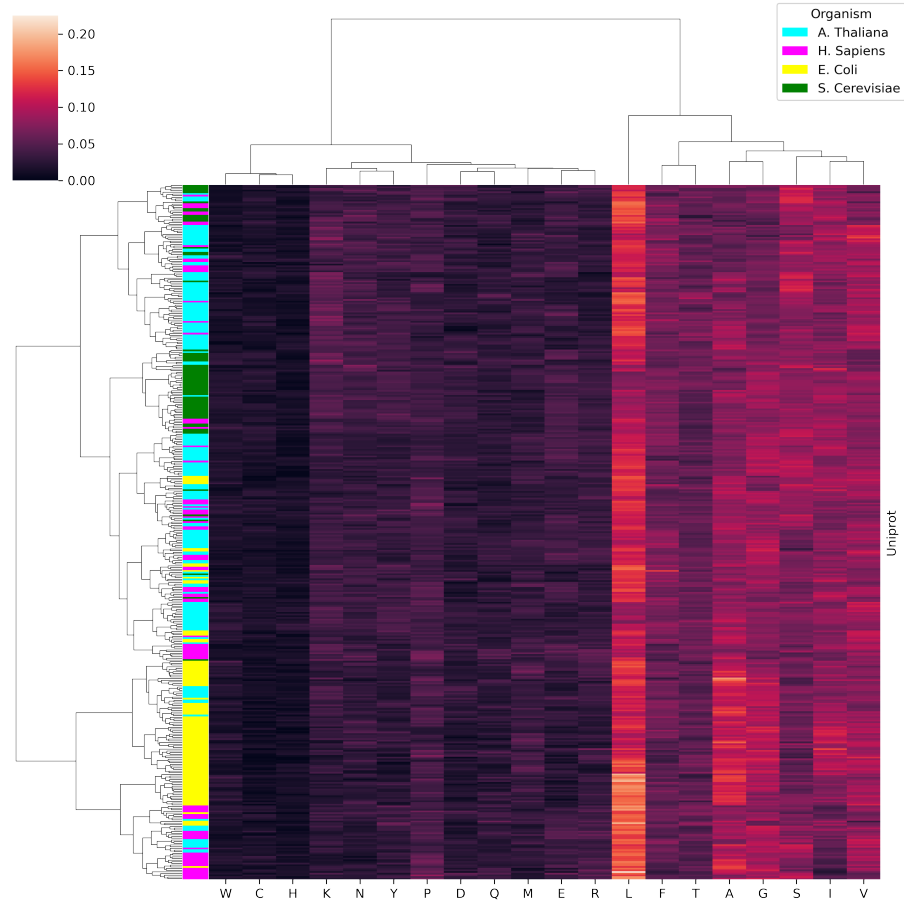


Figure 3.7: Clustered heatmap of all amino acid compositions in the four datasets.

peripheral membrane proteins that are part of active transport complexes but do not transport substrates themselves, as well as channels and pores with beta-barrel structures.

Next, we tested the impact of sequence clustering with cd-hit on the number of samples, and the classification performance (see Figures 3.8 and 3.9). The selection bias was estimated by shuffling the dataset 20 times, each time creating a training set for hyperparameter optimization through cross validation, and an independent test set for evaluation.

Then, we trained a PCA+SVM pipeline on the AAC, PAAC and PSSM features, with the meta-organism dataset, and sequence clustering at a threshold of 70%. The results are close to the average result obtained with models trained on the individual organisms (see Table 3.6). The highest training- and test scores were achieved by the PSSM feature. Since *E. coli* is the phylogenetically most distant organism and forms its own AAC-cluster, we repeated the test without *E. coli* proteins in the dataset. This increased the F1 scores of the PSSM feature by roughly 3%.

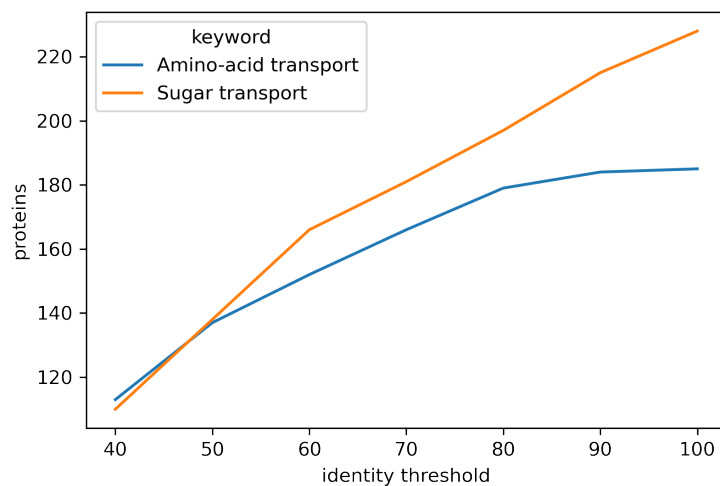


Figure 3.8: Number of amino-acid and sugar transporters in the meta-organism dataset, which contains transporters from *A. thaliana*, *E. coli*, *H. sapiens* and *S. cerevisiae*, at different sequence clustering thresholds.

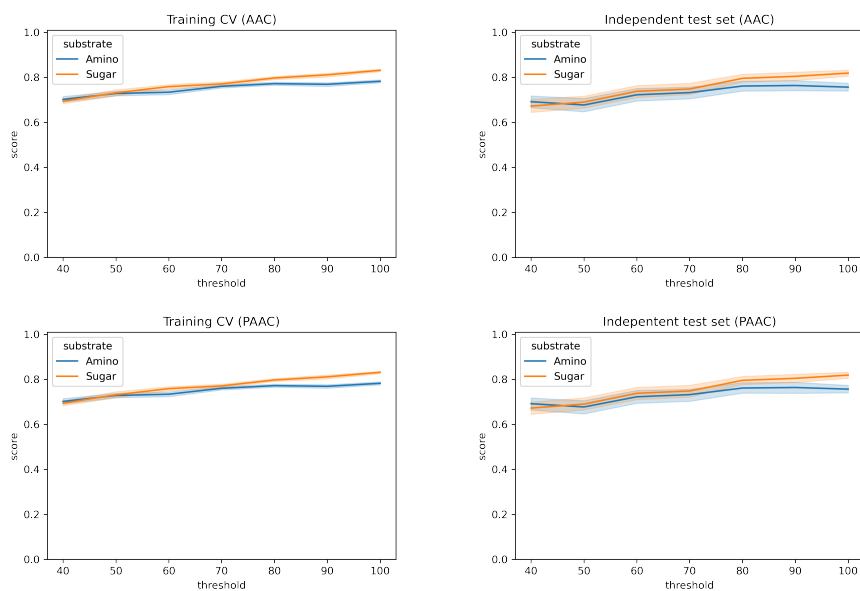


Figure 3.9: Classification performance of the meta-organism dataset at different cd-hit sequence clustering thresholds.

Organisms	Feature	F1 score (mean)	
		training set	ind. test set
All	AAC	0.841	0.743
	PAAC	0.814	0.784
	PSSM	0.935	0.914
	Combined	0.913	0.899
Eukaryotes	AAC	0.785	0.800
	PAAC	0.868	0.778
	PSSM	0.965	0.939
	Combined	0.918	0.899

Table 3.6: Evaluation results when training on a dataset that contains transporters from three or four model organisms.

Class	Dataset	Precision	Recall	F1	Accuracy	Samples
Amino acid	cross validation	0.968	0.957	0.962	0.957	92
	training set	1.000	1.000	1.000	1.000	92
	ind. test set	0.955	0.913	0.933	0.913	23
Sugar	cross validation	0.964	0.972	0.968	0.972	107
	training set	1.000	1.000	1.000	1.000	107
	ind. test set	0.929	0.963	0.945	0.963	27
Macro avg.	cross validation	0.966	0.965	0.965	0.965	199
	training set	1.000	1.000	1.000	1.000	199
	ind. test set	0.942	0.938	0.939	0.940	50

Table 3.7: Comparison between commonly used metrics for evaluating the performance of machine learning models. The eukaryotes-only meta-organism dataset was trained to distinguish between sugar- and amino acid transporters, using the multi-PSSM feature in combination with PCA and a SVM. The accuracy score was calculated directly from the true and false predictions, since the dataset is only slightly imbalanced. The corresponding balanced accuracy score, which takes class imbalance into account and is defined as the arithmetic mean of sensitivity and specificity, was 0.938 for the independent test set.

Finally, we took a closer look at the best-performing meta-organism model, by calculating precision, recall, F1 and accuracy for each individual class (see Table 3.7). The results were consistent across the two classes, with a good balance between precision and recall, as well as between the training scores and test scores. This suggests that the model is not biased towards one of the substrates, and does not overfit to the training data.

3.4.4 Additional substrates

Next, we tested the suitability of our method for multi-label classification tasks, by distinguishing between more than two substrate classes.

To this aim, we created a dataset of *A. thaliana* transmembrane transporters with four different substrate types: *Amino-acid*, *sugar*, *electron* and *potassium*. The substrates were selected according to the number of available samples. We excluded parent categories such as *Protein transport* and *Ion transport*, as the protein sets associated with those terms are too diverse in terms of amino acid composition.

Our transporter dataset contained 84 sugar transporters, 56 electron transporters, 61 potassium transporters, and 40 amino acid transporters. When analyzing the dataset for outliers, we again found the seven GDU family amino acid transporters (see Section 3.4.1). Of the 61 potassium transporters, six were also annotated with *Sodium transport*. These six sodium-potassium pumps were outliers compared to the remaining 55 proteins, and were excluded. Finally, we ended up with 33 amino acid transporters and 55 potassium transporters. The number of sugar and electron transporters did not change.

While AAC and PAAC performed worse for multi-class classification compared to the binary case, the multi-PSSM model and the combined feature managed to produce very similar results. Amino-acid transporters were the weakest class in terms of prediction accuracy, and its F1 scores on the independent test set had the largest variance across the shuffles. This can be explained by the small number of class members: When there are only seven transporters in the independent test set, then one additional false positive can change the accuracy by several percentage points. For this reason, we repeated the evaluation ten times, with different random seeds for choosing the independent test set.

In contrast to binary classification, PCA for dimensionality reduction performed worse than feature selection in the multi-class case. The results are shown in Table 3.8 and Figure 3.10.

3.4.5 Automatic feature combination

In this project, we discuss three different machine learning features: AAC, PAAC, and PSSM. The latter has three parameters that can be adjusted, namely the blast database (Uniref50, Uniref90), the number of iterations, and the E-value threshold for deciding the inclusion of a particular protein in the PSSM. In order to find the best feature, or combination of features, the user would have to try every option individually.

Feature	Proc.	Amino-acid		Electron		Potassium		Sugar		Mean	
		train	test	train	test	train	test	train	test	train	test
AAC	None	0.794	0.697	0.893	0.890	0.964	0.959	0.887	0.863	0.885	0.852
	ANOVA	0.815	0.675	0.892	0.862	0.965	0.955	0.896	0.875	0.892	0.842
	PCA	0.770	0.673	0.875	0.891	0.945	0.959	0.863	0.844	0.863	0.842
PAAC	None	0.750	0.701	0.867	0.892	0.918	0.951	0.885	0.883	0.855	0.857
	ANOVA	0.805	0.734	0.886	0.884	0.925	0.955	0.904	0.884	0.880	0.864
	PCA	0.641	0.647	0.586	0.542	0.784	0.852	0.770	0.790	0.695	0.708
PSSM	None	0.950	0.935	0.983	0.974	0.982	0.974	0.968	0.975	0.971	0.964
	ANOVA	0.953	0.945	0.987	0.958	0.987	0.978	0.975	0.961	0.975	0.960
	PCA	0.990	0.946	0.978	0.969	0.984	0.984	0.985	0.968	0.984	0.967
Combined	None	0.915	0.931	0.972	0.967	0.984	0.984	0.951	0.964	0.956	0.961
	ANOVA	0.942	0.922	0.975	0.960	0.984	0.989	0.969	0.953	0.968	0.956
	PCA	0.911	0.913	0.942	0.945	0.985	0.990	0.945	0.952	0.946	0.950

Table 3.8: Multi-class classification F1 scores for *A. thaliana*, between four substrates. All scores were averaged over ten evaluations, each time with a different random seed.

To help with this, we also implemented an automatic feature combination algorithm, through the scikit-learn transformer interface. For each possible combination of features, the corresponding feature datasets are concatenated and then standardized. Constants for standardization are derived from training data, to avoid information sharing with the test set. After performing a 5-fold cross validation on each combination of features and SVM hyperparameters, the combination of feature datasets and the SVM parameter set that led to the highest average F1 score across the cross validation folds are selected. Finally, the model is tested on an independent test set, to estimate the overfitting.

As a test, we trained a RBF-SVM on the same *A. thaliana* transporter dataset with four substrates that was described in the previous paragraph. For each protein, we calculated the AAC and PAAC features, as well as four different PSSM features, where we performed one or three iterations of PSI-BLAST on either the Uniref50 database or the Uniref90 database. The E-value threshold was set to 0.002 in all cases. For the sake of readability, we define $PSSM(R, i)$ as the PSSM calculated by performing i iterations on database *UnirefR*.

Before optimizing any of the SVM hyper-parameters, the model with automatic feature combination already achieved average F1 scores of 0.968 and 0.936 on the training and test set, respectively. The automatic feature combinator selected the PAAC together with $PSSM(50, 1)$ and $PSSM(50, 3)$. When adding the SVM parameters to the optimization process, the scores improved to 0.974 and 0.976. This time, the algorithm combined the AAC with $PSSM(50, 3)$, $PSSM(90, 1)$ and $PSSM(90, 3)$. The fact that F1 scores on independent test sets are close to the training scores suggests that selecting feature combinations based on training data did not cause considerable overfitting.

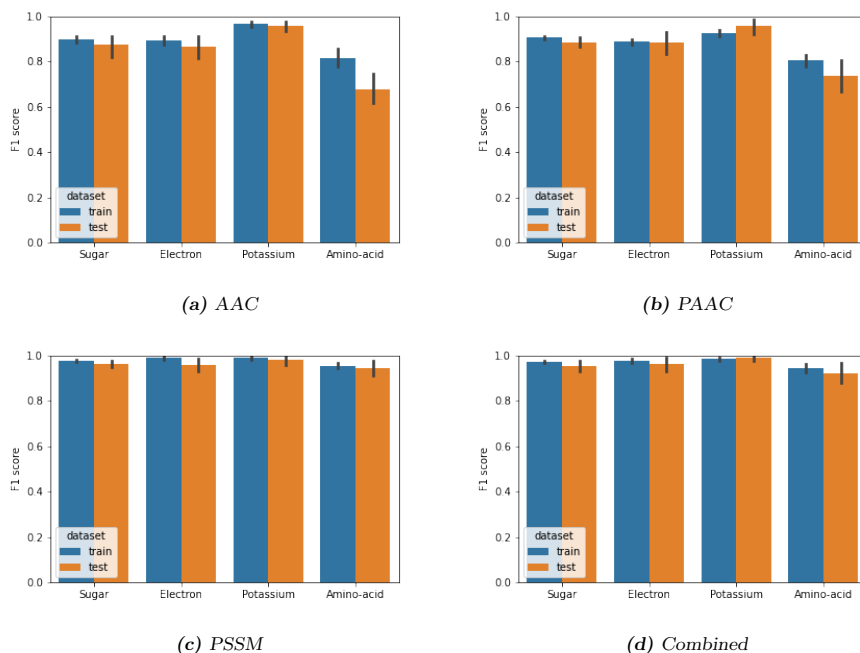


Figure 3.10: Classification results of four-substrate multi-substrate classification with transport proteins from *A. thaliana*. The machine learning pipeline was made up of ANOVA-based feature selection, and a SVM with RBF-kernel. The dataset was split into a training and a test set. A cross validation was performed on the training set to optimize parameters, and the resulting model was applied to the independent test set. The entire process was repeated 10 times, the varying scores between different train/test splits are represented by the standard deviations in the plot.

Finally, we created 1,000 random training and validation sets from the data, in order to estimate how often each individual feature gets selected as part of the best combination. Across the 1,000 training datasets, the model achieved average F1 scores of 0.973 ± 0.01 and 0.967 ± 0.03 on the respective training and validation datasets. The PSSM calculated from three iterations on Uniref50 was selected the most, it was part of the best feature combination in 81.2% of all cases. AAC and PAAC were selected as part of the best combined feature with frequencies of 25.6% and 6.7%. The remaining features, namely $PSSM(90, 3)$, $PSSM(90, 1)$ and $PSSM(50, 1)$ were part of the best feature in 73.1%, 41.5% and 31.9% of cases, respectively. On average, the PSSMs derived from more distantly related proteins seem to produce better models than those comprised of closely related sequences. The combination of features that was selected the most was $(PSSM(50, 3), PSSM(90, 3))$, with a frequency of 15.7%. Overall, 46 out of 62 possible combinations were selected at least once, and 34 of these 46 combined features occurred with a frequency of less than 2%.

3.4.6 Comparisons between classification algorithms

Finally, we compared the performance of the SVM model to that of other classification algorithms. As a benchmark dataset, we again employed the *A. thaliana*

multi-class dataset with four substrates, along with the automatic feature combination algorithm that we described in the previous paragraph. We tested four additional models: Random Forest (RF), K-Nearest Neighbors (KNN), a linear SVM that gets optimized with stochastic gradient descent (SGD-SVM), as well as a single-layer Perceptron. The KNN and RF classifiers are inherently multi-class, whereas the SGD-SVM and Perceptron classifiers employ a one-vs-rest voting scheme to achieve multi-class classification.

For evaluation, we divided the data into a training- and test set, in a stratified fashion. Hyperparameter optimization was performed on the training set, by selecting the parameters that achieved the highest macro-averaged F1 score in a cross validation. Then, the model was then tested for overfitting on the independent test set.

To get a baseline, we first evaluated the RBF-SVM model that was used in the remainder of the study. It achieved average training and test scores of 0.974 and 0.976. The SGD-SVM showed a similar performance, with average F1 values of 0.961 and 0.974 on the training and test sets.

The two multi-output classifiers, RF and KNN, performed slightly worse during the evaluation, with training- and test scores of 0.892 and 0.878, as well as 0.914 and 0.866, respectively. Binary classifiers in combination with one-vs-one or one-vs-rest voting schemes seem to achieve better results on this particular dataset than classifiers that are inherently multi-class.

The Perceptron model with default parameters gave F1 scores of 0.972 and 0.974 on the training- and test set, respectively. With elastic net regularization, the F1 scores improved to 0.988 and 1.0. We repeated the entire process 10 times with the same set of hyperparameters to choose from, using different random seeds for creating the independent test set. The results stayed consistent, with average training and test scores of 0.978 ± 0.006 , and 0.969 ± 0.024 . While the Perceptron implementation in scikit-learn is limited to a single output node, its good results could be an indication that more complex neural network architectures will allow for even more accurate models with a higher number of classes in the future.

3.5 Discussion

3.5.0.1 Model creation

At first, we studied a simple, binary classification task between sugar transporters and amino acid transporters in *A. thaliana*, with the goal of taking a close look at the proteins in the dataset, and more specifically their amino acid compositions. We found that substrate specificity depends on a multitude of amino acid types, which are not among the most frequently occurring residues in the sequences (see Figure 3.6). Models have to incorporate information about multiple amino acids in order to reach a good classification performance. The best results were achieved by performing a PCA, and selecting a number of principal components that explain between 80% and 99% of the variance in the AAC dataset. Encoding the sequence through more complex means, i.e. the PAAC and the multi-PSSM feature, further improved the class separation.

We created our transporter datasets from Swiss-Prot, the manually curated subset of UniProt. Proteins were filtered according to their functions based on the UniProt keywords they are annotated with. Only keeping proteins annotated with the *Transmembrane* keyword in the dataset proved useful in removing peripheral membrane proteins, which are associated with transmembrane transport complexes. These proteins represent outliers, since they do not carry out the actual transport themselves. Gene ontology terms related to the transport of a specific substrate across a membrane are available as well, but fewer proteins are annotated with them.

Next, we created a sugar/amino acid dataset for three additional organisms, and compared the transport proteins, the amino acid compositions, as well as the classification results (see Tables 3.2 and 3.3). A small set of outliers in terms of AAC was found in *E. coli*, encompassing five proteins with beta-barrel structure. Generally, in terms of three-dimensional structure, there are two shapes of membrane transporters: beta-barrels, which mostly occur in mitochondria, chloroplasts and gram-negative bacteria, as well as alpha-helical bundles, which are more commonly found in cell membranes, and in the endoplasmatic reticulum [374]. We removed porins in the outer membrane of *E. coli* by filtering for the respective keywords, as they were the only transporters with a beta-barrel structure in the sugar/amino acid dataset. Other membranes known for containing beta-barrel transporters are those of chloroplasts and mitochondria, which is something to be aware of when including proteins from those membranes into a dataset.

When comparing the results between organisms, it became apparent that the number of proteins in the dataset with a particular substrate was correlating with the average F1 score of that substrate during cross validation. Via a test on data from *A. thaliana*, we found that the number of substrates per class should be at least 20 (see Figure 3.4). This number also seemed to apply to other organisms. Especially *S. cerevisiae* had problems with overfitting, while also having the lowest sample count of 17 sugar transporters in the dataset (see Table 3.1). Changing the evaluation method to a nested leave-one-out cross validation, which retains more training data for training, improved the classification results and solved the problems with overfitting on the training data (see Table 3.3).

When comparing proteins from different membranes, we did not find considerable differences in amino acid composition between the proteins in our datasets, other than the residues leading to their substrate specificity. This could also be due to the fact that all remaining sugar and amino acid transporters had the alpha helical bundle structure. To test for differences in how the substrate specificity manifests, we trained a model on one organism, and applied it to another. This did not work well for any combination of organisms. Filtering for one organism led to overfitting in all cases, and was caused by the position of the SVM hyperplane, which is derived from the respective training organism (see Tables 3.4 and 3.5).

The meta-organism dataset then included transporters from multiple organisms at the same time, meaning that training and test sets contained transporters from more than one organism. This led to improved results for organisms with a low sample count. Mixing membrane transport proteins from eukaryotes with

proteins from the inner membrane of a gram negative bacterium did not influence the results notably (see Table 3.6). The meta-organism dataset had higher levels of redundancy than the datasets for the individual organisms (see Figure 3.8), and reducing redundancy had a slightly higher impact on classification performance than for the individual organisms (see Figure 3.9).

Finally, we increased the number of substrates to four, and tested the classification performance in *A. thaliana*. Substrate specificity among all ion transporters was too diverse to consider them as one uniform substrate class. This was still the case when removing transporters with beta-barrel structures. Instead we selected Potassium, the ion with the largest number of samples available. As a fourth substrate we included electron transporters, which the TCDB lists as having their own, distinctive transporter mechanism. During the evaluation, the label with the fewest samples, that of amino acid transporters, had problems with overfitting on our ML pipeline when only using the AAC and PAAC as features. The multi-PSSM feature and the combined feature resolved this problem and gave F1 scores above 0.90 for all substrates (see Figure 3.10). In the section 3.4.5, we then used the *A. thaliana* multi-substrate dataset to implement and test an automatic feature combination algorithm, which selects the best combination of feature datasets based on the respective training data. This improved the average F1 scores to 0.974 and 0.976 on the training- and validation datasets, respectively.

In the future, we plan to increase the number of substrates in our models even further. Additional substrate information from molecular function GO terms, and ChEBI [292] annotations on TCDB could lead to a slightly increased sample count. Carefully assembled multi-organism, multi-substrate training datasets will likely enable one to predict substrates in a wide variety of organisms and proteins. Using probabilistic models and multi-label training data are further possibilities as well.

3.5.0.2 Related work

Schaadt and Helms [39][366] created a transporter dataset from the ARAMEMNON database [253], and categorized the proteins into four classes: amino acids, oligopeptides, phosphates, and hexoses. This approach led to good accuracy scores of up to 90%, but was limited to proteins from *A. thaliana*. Mishra et al. [40] then compiled a dataset of membrane transport proteins from UniProt, and assigned them into seven classes: *amino acid*, *anion*, *cation*, *electron*, *protein/mRNA*, *sugar* and *other*. The sequences were clustered at a sequence similarity threshold of 70%. The proteins for the evaluation dataset and the independent test set were then manually filtered, possibly to remove outliers. The same dataset was used subsequently by Alballa et al. [367] in their study. This final dataset was only available in its clustered form, which did not allow for the possibility of creating a single-organism subset. Also, the manual selection process was not further specified. For these reasons, we decided to create a transporter dataset pipeline, with a solid foundation in membrane biology and proteomics. This enables the creation of single- or multi-organism models. Single organism models allow for additional types of ML features in the future, such as the feature based on coexpression networks and genomic neighborhood proposed by Tran et al. [375].

3.5.0.3 Choice of algorithms

Among different machine learning algorithms that we tested, SVMs were among the best-performing classifiers in all test cases, where we tested the substrate prediction of membrane transporters. This makes sense, considering that SVMs are well suited for training datasets with few samples and many features, as long as there are no obvious outliers in the dataset. Other classifiers we tried include random forests (RF), k-nearest neighbors (KNN), a single-layer Perceptron and a linear SVM optimized with stochastic gradient descent (SGD-SVM). When comparing the five algorithms performance on the *A. thaliana* dataset with four substrates and automatic feature combination, the two multi-output classifiers (RF and KNN) were outperformed by binary classification algorithms that employ one-vs-rest voting schemes. The SGD-SVM and the Perceptron performed similarly to the RBF-SVM we used throughout the rest of the project. Employment of custom ensemble classifiers, or more sophisticated neural network architectures, could improve the classification performance even further, possibly at the cost of increased runtime and complexity, or losing the ability to understand the results. For binary classification tasks, the reduction of feature dimensions through principal component analysis improved the model fit in most cases. For multi-class separation, ML pipelines involving feature selection based on ANOVA F-scores outperformed PCA-based models.

3.5.0.4 Conclusions

We found that data visualization, and paying close attention to the biological properties of proteins, allows for the creation of better training datasets, through the discovery and removal of outliers. The most important signifier for labeling a protein as an outlier was the three-dimensional structure, which is now available for a wide variety of proteins thanks to the AlphaFold project [274] and the Protein Data Bank [264]. For example, we found a small subset of five beta-barrel proteins in our *E. coli* sugar/amino acid transporter dataset, which was otherwise made up of proteins with the alpha-helical bundle structure. We also found single-pass membrane proteins and peripheral membrane proteins through their structure, and removed them from the dataset. Creating ML models for single organisms has two main advantages. It allows for a better model fit and predictive accuracy, and for the employment of additional data for creating ML features, such as genomics and transcriptomics data. However, the data availability appears to be too low for most organisms and substrates. We determined the smallest recommended number of transporter samples for each substrate to be at around 20. Using a meta-organism dataset, we were able to train robust SVM classifiers that achieved F1-scores above 0.91 in a 2-class separation task.

3.6 Data and Software Availability

The source code, as well as instructions recreating the development environment, are available at <https://github.com/adenger/subpred3>. The repository also contains instructions for downloading the most recent versions of the raw datasets. A link to the datasets we used for our analyses, which were downloaded on November 15th, 2021, is provided as well.

Chapter 4

Identifying optimal substrate classes of membrane transporters

The contents of this chapter were published in the journal **PLOS One**, under the title *Identifying Optimal Substrate Classes of Membrane Transporters* [376]. Links to the raw data, along with all the code that was implemented, is available at <https://github.com/adenger/subpred4>.

4.1 Abstract

Membrane transporters are responsible for moving a wide variety of molecules across biological membranes, making them integral to key biological pathways in all organisms. Identifying all membrane transporters within a (meta-)proteome, along with their specific substrates, provides important information for various research fields, including biotechnology, pharmacology, and metabolomics. Protein datasets are frequently annotated with thousands of molecular functions that form complex networks, often with partial or full redundancy and hierarchical relationships. This complexity, along with the low sample count for more specific functions, makes them unsuitable as classes for supervised learning methods, meaning that the creation of an optimal subset of annotations is required. However, selection of this subset requires extensive manual effort, along with knowledge about the biology behind the respective functions. Here, we present an automated pipeline to address this problem. Unlike previous approaches for reducing redundancy in GO datasets, we employ machine learning to identify a subset of functional annotations in a training dataset. Classes in the resulting predictive model meet four essential criteria: sufficient sample size for training predictive models, minimal redundancy, strong class separability, and relevance to substrate transport. Furthermore, we implemented a pipeline for creating training datasets of transmembrane transporters that cover a wide range of organisms, including plants, bacteria, mammals, and single-cell eukaryotes. For a dataset containing 98.1% of transporters from *S. cerevisiae*, the

pipeline automatically reduced the number of functional annotations from 287 to 11 GO terms that could be classified with a median pairwise F1 score of 0.87 ± 0.16 . For a meta-organism dataset containing 96% of all transport proteins from *S. cerevisiae*, *A. thaliana*, *E. coli* and human, the number of classes was reduced from 695 to 49, with a median F1 score of 0.92 ± 0.10 between pairs of GO terms. When lowering the percentage of covered proteins down to 67%, the pipeline found a subset of 30 GO terms with a median F1 score of 0.95 ± 0.06 .

4.2 Introduction

Proteins fulfill a variety of functions in each cell. One of these functions is the transport of molecules across biological membranes, specifically those molecules that cannot diffuse across the membrane on their own. There exist dedicated transmembrane proteins, so-called membrane transporters, that enable the uptake of ions and polar molecules such as sugars and amino acids, enable cell communication and signaling, or mediate drug efflux leading to resistance against antibiotics in pathogenic micro-organisms, for example [377].

Currently, the substrates transported by many of these transporters are still unknown [233], and closing the sequence-annotation gap is an important task in bioinformatics [378]. Whereas certain transmembrane transport proteins can be assigned to their substrate class based on homology to a protein family, the exact substrate is often determined by a very small part of the sequence, while the remaining structure is an indicator of the transport mechanism, the protein family and the ancestry, respectively [379] [365] [26]. On top of that, large parts of known proteomes do not possess enough sequence similarity to related proteins to infer functional information from, and thus belong to the so-called *dark proteome* [18]. The ability to identify transporters for specific substrates within a (meta-)proteome is essential across various research fields, including biotechnology [16][14][15], pharmacology [7][5][6], and metabolomics [8][9].

Machine learning has been successfully applied to many fields [21], including a number of studies that aimed to assign substrate classes to membrane transporters. In 2010, Schaadt and Helms [39] trained Support Vector Machines (SVMs) on the frequencies of amino acid types in protein sequences, in order to distinguish between four substrate classes in *A. thaliana*. Other studies used sequence encodings based on multiple sequence alignments [366], position-specific scoring matrices (PSSMs) [370], and physicochemical properties of the amino acids in the sequence [40]. Text embedding methods adapted from Natural Language Processing (NLP) were also employed for protein annotation [378]. Previously [349], we introduced a machine learning approach that combined amino acid frequencies with different PSSMs for each protein, and optimized the feature dimensions through feature selection on the training dataset. This model achieved high evaluation scores on a dataset of transporters from *A. thaliana*, and performed even better when applied to a meta-organism dataset.

Machine learning classifiers achieve optimal performance when trained on homogeneous, well-separated classes, as this improves pattern recognition, reduces model complexity, and shortens training time [380]. However, the abundance

of functional annotations and the complexity of their ontologies make it challenging to define classes of proteins that are distinct, while also providing useful information in their predictions [381]. In previous studies, classes for predictive models that aim to annotate substrates for all membrane transporters in a proteome were selected manually. Schaadt and Helms [39] considered transporters belonging to four classes of substrates in *A. thaliana* (amino acids, oligopeptides, phosphates, and hexoses). Mishra et al. [40] split the transporters available in Uniprot (version 2013_03) into seven substrate classes: amino acid/oligopeptide, anion, cation, electron, protein/mRNA, sugar and *other*, and removed any transport protein annotated with more than one substrate. The TranCEP method [367] used this dataset for training as well. In our recent study [349], we considered four substrate classes (sugar, amino acids, potassium and electron).

However, relying on manually defined substrate classes has inherent limitations. Broad categories like *cation* may merge proteins with diverse sequences, structural folds, and transport mechanisms, potentially introducing confounding variables that prevent accurate predictions. Likewise, the choice of very specific substrate classes, such as *magnesium*, might not contain enough samples per class for training reliable machine learning models [349]. Selecting optimal substrate classes for each dataset requires extensive manual effort and biological knowledge. Therefore, we present an automated pipeline that generates distinct, homogeneous protein classes, using a greedy algorithm that optimizes class selection based on training data. Furthermore, the automated pipeline can rapidly compile convenient datasets for selecting substrate classes and training ML models across a variety of organisms, including bacteria, plants, mammals and yeast, and combinations of those.

Although methods for reducing redundancy in sets of GO terms were previously developed [382][381], they were not optimized for machine learning tasks. Instead, these approaches were primarily designed for gene set enrichment analysis, operating on the entire GO and gene dataset. Therefore, they lack customization for specific datasets and tasks, such as the ability to identify annotations of transmembrane transporters that are both associated with a sufficient number of proteins for training and specific enough to provide meaningful information about the function of each transporter that is classified with the model.

With the recent rise of transformer models and natural language processing, first attempts have been made to adapt these methods to general protein function prediction [41]. Whereas most of these methods aim to assign annotations from the entire Gene Ontology (GO) [47] to the proteins, creating a small and non-redundant subset of functional annotations would considerably reduce the number of nodes in the output layer, leading to simpler models that, in practice, may provide a similar amount of information by their predictions.

Thus, the aim of this study was to implement a method that identifies an optimal set of transport-function-related GO terms, so that putative transporters can be reliably annotated with one or more of these terms. To this end, we implemented an algorithm that drastically reduces the number of GO terms a protein dataset is annotated with, and then returns a non-redundant subset with highly accurate pairwise classification results, that still contains the original functional information found in the complete set of GO annotations. Additionally, we introduce an improved software pipeline to compile protein datasets, along with functional

annotations, that allows for filtering of transporter sequences by various measures, indicating data quality and other attributes. Unlike prior approaches that exclude multi-substrate proteins [39][40][349], our pipeline supports multi-label datasets, allowing a single protein to belong to multiple classes. This automated pipeline enables rapid and accurate identification of biologically relevant, separable transporter classes in a protein dataset, significantly reducing manual curation efforts.

4.3 Materials and methods

4.3.1 Data retrieval and preprocessing

Protein data was obtained from UniprotKB, version 2022_05 [360]. Proteins with fragmented sequences or without experimentally proven existence were not included in the dataset. After this initial filtering, a total of 1,693,707 proteins remained in the dataset. Non-standard amino acid codes, such as B, Z, J or X, were removed from the sequences. These amino acid descriptors were found in 45,718 protein sequences, although none of them were part of the membrane transporter datasets we explored in the results section.

The corresponding dataset 2022-11-03 of all Gene Ontology (GO) annotations [47], including those that were electronically inferred from other sources, was downloaded from the official GO webserver and filtered for those genes with Uniprot annotations. The relations between individual GO terms were downloaded from the Gene Ontology website, in the OBO format.

The code, along with links for downloading the raw data, is provided in the official repository (see Appendix C.1).

4.3.2 Dataset creation pipeline

A dataset pipeline was implemented that creates filtered protein datasets and annotates the proteins with a subset of GO terms. The protein sequences are first filtered for a set of organisms, while the GO terms can be filtered for those that are ancestors of a particular node (e.g. *transmembrane transporter activity*). The proteins in the filtered Uniprot dataset are then annotated with the GO annotations from Section 4.3.1. Outdated GO terms are updated to their current version, using the *alt_id* node annotations in the GO graph. If a protein is associated with a particular GO term, then the ancestors of that GO term according to the filtered GO graph receive that relation as well. The exact parameters of the pipeline are listed in Appendix C.3.

For the transmembrane transporter dataset, only *molecular_function* GO terms that are descendants of *transmembrane transporter activity* (abbreviated as TTA) were kept in the dataset, and only edges with the *is-a* relation were allowed. Protein annotations were filtered for the *enables* qualifier, which denotes that the protein is directly responsible for the molecular function, as opposed to acting upstream, for example.

4.3.3 Pairwise similarity scores

After creating a dataset of n proteins, and annotating them with a subset of m GO terms, we used the protein sequences of the annotated proteins, and other associated data, to create similarity matrices of size $m \times m$ between pairs of GO terms, with the goal of comparing these similarity scores to each other and to evaluation scores calculated from machine learning models.

4.3.3.1 Sequence similarity

In order to determine a “sequence similarity between two GO terms”, a matrix of pairwise sequence similarity scores between all proteins annotated with these two GO terms was computed. This matrix was then aggregated into a single number by taking the average, minimal or maximal value.

Pairwise sequence similarity scores were calculated with the Biostrings R-package, using the Needleman-Wunsch algorithm [311] for optimal global sequence alignments, with the BLOSUM62 substitution matrix [68] and gap opening- and gap extension penalties of 10 and 4, respectively.

4.3.3.2 Semantic similarity

Semantic similarity scores between pairs of GO terms were calculated using the GOSemSim R-package [383]. Here, we used the method proposed by Wang et al. [384], as it can be calculated directly from the GO graph, and avoids the drawbacks of semantic similarity measures that are calculated on term-based information content [384, 385, 386].

4.3.3.3 Overlap matrix

The overlap matrix for a set of m GO terms contains the number of proteins two GO terms have in common. Each position $i, j < m$ in the matrix corresponds to the size of the intersection set between the set of proteins annotated with GO term i , and the set of proteins annotated with GO term j .

4.3.3.4 Chemical similarity between substrates

Cross-ontology relations between GO and the ChEBI ontology [292] were retrieved through the QuickGO API [289]. After the preprocessing described in Section 4.3.2, the GO terms are mapped to their corresponding ChEBI terms, if any are available. Relations between ChEBI and GO terms were filtered for “has_primary_input” and “has_participant”, denoting a direct connection. Ancestors of annotated ChEBI terms were associated with the respective GO terms as well, according to a filtered ChEBI graph that only contained “is_a” relations.

In order to calculate a measure of the chemical similarity between two substrate transport GO terms, the GO annotations are first filtered for those that are annotated with ChEBI identifiers representing primary substrate molecules. Molecular fingerprints are derived from the SMILES-representations [387] of the ChEBI terms, using RDKit [388]. Four different methods are used: Morgan [389], Atompairs [390], Topological torsion [391], and MACCS [392]. The chem-

ical similarity between two substrates is estimated as the Tanimoto coefficient [393] of their fingerprints.

The chemical similarity between two substrate-transport GO terms is then defined as the Tanimoto coefficient between the respective transported substrates. If more than one molecule is associated with a GO term, the average, maximal and minimal pairwise Tanimoto coefficient is calculated.

4.3.3.5 Evaluation of pairwise machine learning models

Machine learning (ML) models were trained to distinguish sets of protein sequences annotated with either one of two different GO terms.

To this aim, the protein sequences were transformed into numerical vectors of constant length by applying the feature generation algorithms described in [349] and Chapter 3: Amino acid composition (AAC), pair amino acid composition (PAAC), and four PSSM features. The individual feature vectors were individually standardized along the sample axis, and concatenated into a single vector of length 1,600. The four PSSMs for each protein were calculated by calling PSIBLAST with different parameters: either one or three iterations of PSIBLAST, and either Uniref50 or Uniref90 [394] as the BLAST database, respectively. For this analysis, we did not cluster the dataset at a sequence identity threshold of 70% as we did before [349], since we wanted to estimate the correlation between sequence similarity and the performance of the machine learning models.

Class labels were derived from the GO annotation data described in Section 4.3.2. Depending on what type of model was used, all GO terms annotated to fewer than k or k_u proteins were removed from the dataset.

In our previous work [349], we observed that SVMs with RBF kernels were usually among the best-performing models, as long as the sample size was greater than 20, and no clear outliers could be found in the training dataset. While we previously removed multi-substrate transporters if both substrates were part of the classification task, we now used multi-output SVM classifiers that can be trained on a dataset where one sample can belong to multiple classes (e.g. when distinguishing proton transporters from proton+molecule symporters or antiporters). In the latter case, only pairs of GO terms were allowed where each term is associated with at least $k_u \leq k$ unique proteins in the dataset, excluding those that were part of the intersection. This was done to avoid cases where all or most proteins that are annotated with the first GO term are also annotated with the second GO term, which can, for example, occur when the former is the parent node of the latter.

Since the feature dataset consisted of 1,600 feature dimensions, but only hundreds or tens of samples, we added a feature selection algorithm to the machine learning pipeline. Based on the training dataset, the percentage p of best feature dimensions was selected, based on the ANOVA F-score between the respective feature dimension and the corresponding class label.

The performance of each model was evaluated by computing precision, recall and F1 scores individually for each class. The metrics represent the average score across five iterations of a nested cross validation, where the outer cross validation splits the data into a training dataset and an independent test set. An

inner cross validation on the training dataset optimizes the hyper parameters of the model, that is then evaluated on the independent test set. The position i, j in the resulting matrix of pairwise evaluation metrics corresponds to the average F1 score on the independent test sets across all outer folds, when distinguishing the positive class i from the negative class j . The macro-averaged F1 score for a pair of GO terms was taken as the arithmetic mean between positions i, j and j, i .

4.3.4 Greedy algorithm for clustering of GO terms

GO terms can be closely related in terms of function, and therefore have many annotated proteins in common. Trying to distinguish two very similar GO terms from each other through means of machine learning can be a challenge, since the small number of unique samples available for each class makes it harder for the ML model to find the correct patterns in the feature vectors, and since the function may be too similar. For that reason, we implemented a pipeline that reduces the redundancy in a subset of GO terms.

First, all GO terms with fewer than n associated proteins were removed from the dataset, since they would anyhow not possess enough samples available for training a ML model. In some cases, we also removed the top p th percentile of GO terms according to the distribution of sample counts. For the remaining GO terms, a sparse matrix of pairwise ML evaluation scores was created. For each pair of GO terms, a multi-output SVM with ANOVA-based feature selection was evaluated by a 5-fold cross-validation approach. The average F1 score across the respective test sets of CV iterations was then used as a measure for how well the model can assign the correct GO terms to the associated protein sets, using the ML features described in Section 4.3.3.5. Evaluation scores were only calculated for pairs of GO terms where both have at least m unique proteins that are not annotated with the respective other GO term. If not enough unique proteins were available for training a classifier on a pair of GO terms, the respective position in the matrix was left empty. This threshold m was necessary to avoid cases where the annotated proteins of one GO term are a subset of the annotated proteins of another GO term, which would leave us with little or no training data for the minority class.

The set of GO terms was then optimized via a greedy algorithm that iteratively removed one GO term per step. For each GO term g in the subset, the coverage of remaining GO terms (i.e. those that have not been removed yet and are not equal to g) was calculated by dividing the number of proteins annotated with those GO terms by the total number of proteins. If removing a particular GO term would not cause the coverage to fall below a specified threshold, that GO term was considered as a candidate for removal. For each of these GO terms, the potential impact (increase) of its removal on the average F1 score was calculated. When calculating the average, the empty positions in the F1 matrix were filled with a specified constant value, by default with -1, which ensures that these GO terms get lower average scores and are removed first. The list of GO terms considered for removal was then shortened to those with the highest positive impact on the average F1 score when deleted, plus a small term ϵ , which causes GO terms with a very similar impact on average F1 scores, e.g. a difference of less than 0.001, to be considered to have the same impact.

The GO terms with the highest positive impact were sorted by their level in the GO graph (i.e. the distance from the selected root node), and either the most abstract or the most concrete GO term was selected for removal, according to the preference of the user. Should there still be multiple GO terms on the same level, one of those terms was drawn by a deterministic random number generator and removed from the set of GO terms. This method was repeated, until no more GO terms could be removed without the coverage falling below the specified threshold.

The exact algorithm used for optimizing GO annotation sets is described in more detail in the supplemental material (see Appendix C.2).

4.3.5 Pipeline overview and complexity

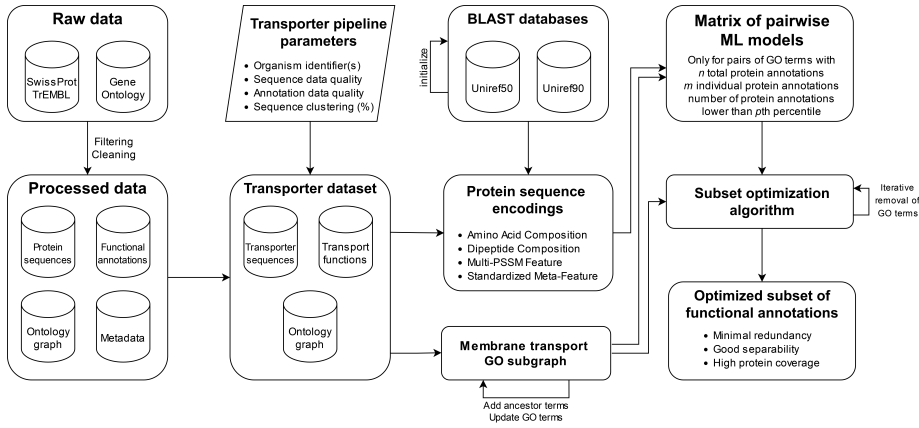


Figure 4.1: Flowchart showing the entire method pipeline. First, the raw data is filtered and cleaned, and converted to a binary data format for faster reading (see Section 4.3.1). Next, the general protein dataset is converted to a specific transporter dataset according to specified parameters (see Section 4.3.2). Then, the protein feature generation algorithms described in our previous study [349] are applied to the data, and pairwise ML models are trained and evaluated (see Section 4.3.3.5). Finally, the iterative optimization algorithm described in Section 4.3.4 and in C.2 is applied to the dataset, and an optimized set of functional annotations is returned.

The full pipeline, including transporter dataset creation, feature calculation, pairwise machine learning model evaluation, and application of the greedy algorithm, is outlined in a flowchart available in Figure 4.1. The entire software package makes extensive use of caching, so after an initial pre-calculation phase that requires up to a few hours on a modern desktop processor, the entire pipeline can subsequently be executed within seconds. Pre-calculated PSSMs for the feature calculation are provided together with the raw data, as the calculation of four PSSMs for each protein is the most time-intensive task. During the pre-calculation step for the greedy algorithm (see Section 4.3.3.5), a total of $\frac{k^2-k}{2} - j$ ML pipelines are trained and evaluated, where k is the initial number of GO terms and j is the number of GO term pairs that share too many proteins to have enough samples per class for training (e.g. when two annotations are parent and child nodes in the Gene Ontology graph). The greedy algorithm

itself removes GO terms until a number of conditions are met, meaning that its linear runtime has an upper bound of k .

4.4 Results

4.4.1 Dataset creation pipeline

First, we employed an updated version of the transporter dataset creation pipeline described in reference [349] to create a meta-organism transmembrane transporter dataset. This dataset included transporters from four organisms: *Arabidopsis thaliana*, *Saccharomyces cerevisiae*, *Escherichia coli* and human.

The proteins were filtered according to multiple criteria that measure the quality of the underlying data: The manual review status by Swissprot reviewers, whether there are known genes associated with the proteins, the type of evidence used to annotate the protein with a particular GO term, and whether the protein is still part of the dataset after applying sequence clustering with a particular threshold (see Table C.1). Table C.2 shows the number of transport-related GO terms that these proteins are annotated with.

When selecting only manually curated data based on experiments, we obtained a total of 1.602 transporters at a clustering threshold of 70%. With all of Uniprot and the same parameters, we would only get additional 250 proteins, meaning that these four organisms provide an adequate representation of all organisms in Uniprot. The transporters in this meta-organism dataset were annotated with 679 transmembrane-transporter-related GO terms, compared to 639 for all of Uniprot. This was due to the sequence clustering, where lower clustering thresholds are more likely to assign two proteins with different GO terms into the same cluster, thereby potentially removing the GO terms of one of these proteins from the dataset.

4.4.2 Application of the redundancy reduction pipeline to the meta-organism dataset

Next, the GO term clustering pipeline described in Section 4.3.4 was applied to this dataset. Due to the large number of proteins, we found it necessary to not only remove GO terms with fewer than 20 proteins from the dataset, but to also remove the top p th percentile of most general GO terms, according to the distribution of protein counts. Protein counts follow a steep exponential distribution, with a median of 4.0 and a maximum of 815, excluding the root node *transmembrane transporter activity* (see Figure 4.2). When keeping GO terms at or below the 99th percentile, which lies at 454.44, only seven out of 695 GO terms are removed. Removing the 35 terms above the 95th percentile at 110.6 leaves us with 660 GO terms.

Before applying the greedy algorithm and when calculating pairwise F1 scores with $m = 10$ and after removing GO terms with less than $n = 20$ proteins, the dataset consisted of 1661 proteins annotated with 137 GO terms. For 1.666 pairs of these GO terms, no evaluation scores were available since their protein sets were too similar to train a ML algorithm on them. The median F1 score for the remaining pairs was 0.916.

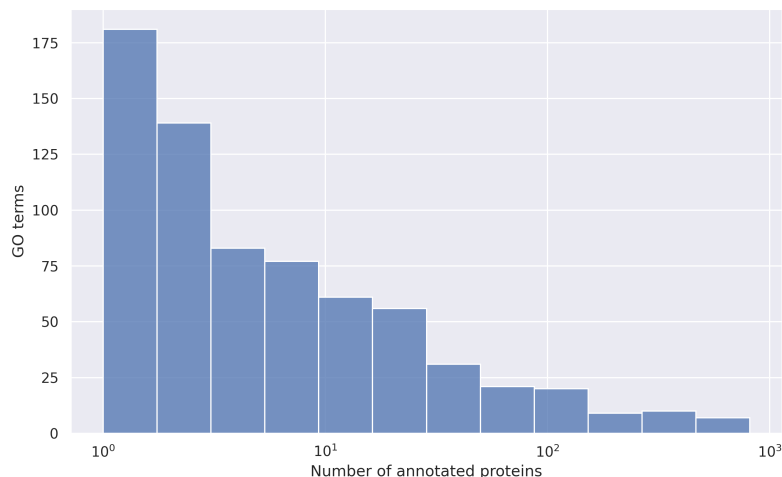


Figure 4.2: Number of annotated proteins for molecular function GO terms related to transmembrane substrate transport. The dataset only contains SwissProt proteins and GO term annotations based on experimental data.

With $p = 0$ (i.e. when not removing the top p -th percentile GO terms according to the number of annotated proteins), the median F1 score at a coverage of 100% starts at around 0.845 (see Figure C.1). Subset sizes are comparatively small (see C.2), many pairs without F1 scores still remain in the dataset (see Figure C.3). This is due to the presence of abstract GO terms that cover many proteins, but also have large overlaps with each other, such as *channel activity* or *secondary active transmembrane transporter activity* (see Figure C.4). After the greedy algorithm removed these abstract terms, they were replaced by their child terms, which are more precise and have less overlap. This more than doubled the number of GO terms needed to reach the required coverage, and caused the number of pairs without scores to drop, while the median score rose sharply. At lower coverage thresholds, the subset sizes dropped again, while the F1 scores and available pairs continued to improve.

Removing the top 5th percentile of GO terms at the beginning of the pipeline with $p = 5$ means that they do not have to be removed by the greedy method, and already yields good results at a coverage threshold of 100% (see Figures 4.3, 4.4 and 4.5). With $m = 5$ and a coverage threshold of 0.96, which was the highest coverage for which there were F1 scores available for all pairs, we found a subset of 49 GO terms, with a median pairwise F1 score of 0.92 ± 0.10 . Overlaps between the terms were low, the mean overlap was 2% and the maximum overlap was 77% (see Figure 4.6).

Finally, we calculated the best subsets with $m = 20$. At this number of unique proteins per GO term, the highest possible coverage is 67% with $p = 5$, and 60% with $p = 0$ (see Figures 4.5 and C.3). The resulting subset of length 30 reached a median F1 score of 0.95 ± 0.06 . Overlaps were minimal (see Figure C.5), with a mean and maximum overlap of 1% and 42%, respectively. In conclusion,

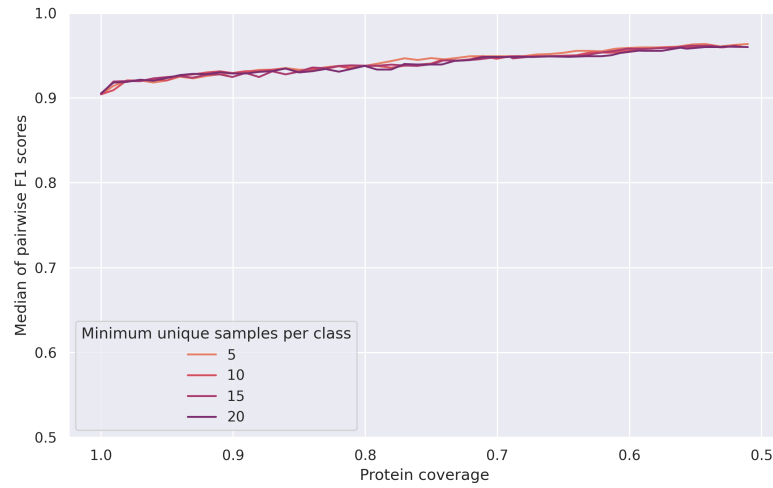


Figure 4.3: Median F1 scores between pairs of GO terms for the meta-organism dataset, after removing the top 5th percentile of GO terms according to sample count.

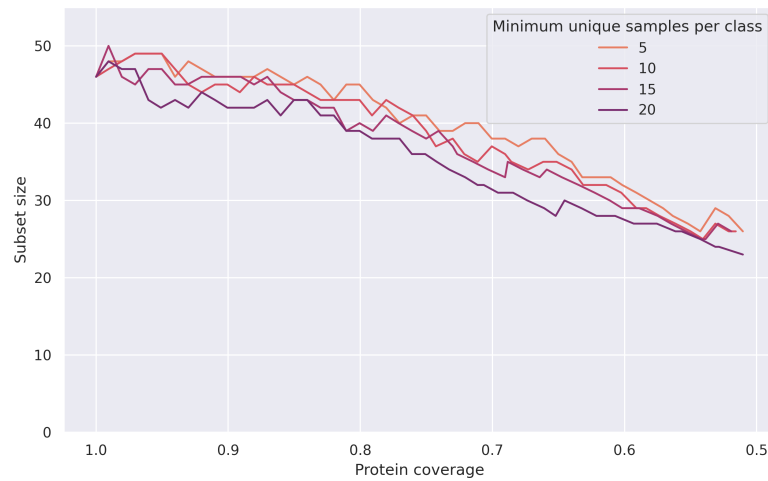


Figure 4.4: Final subset sizes for the meta-organism dataset generated by the pipeline, after removing the top 5th percentile of GO terms according to sample count.

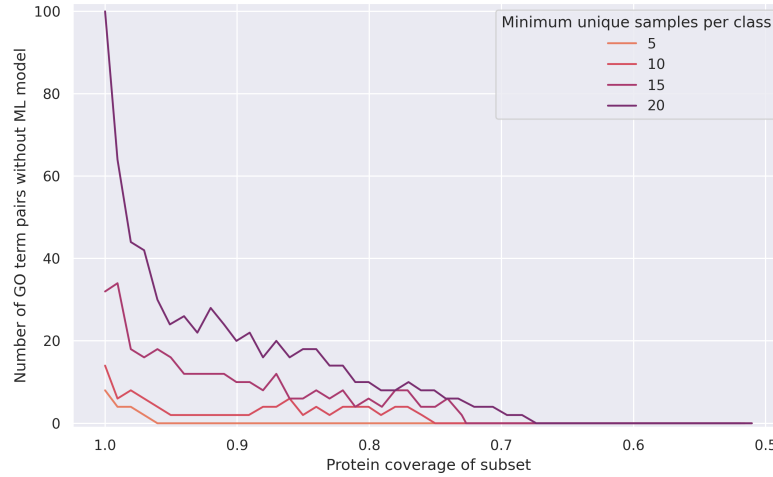


Figure 4.5: Number of GO term pairs without F1 scores for the meta-organism dataset due to a unique sample count of less than m , after removing the top 5th percentile of GO terms according to the sample count distribution.

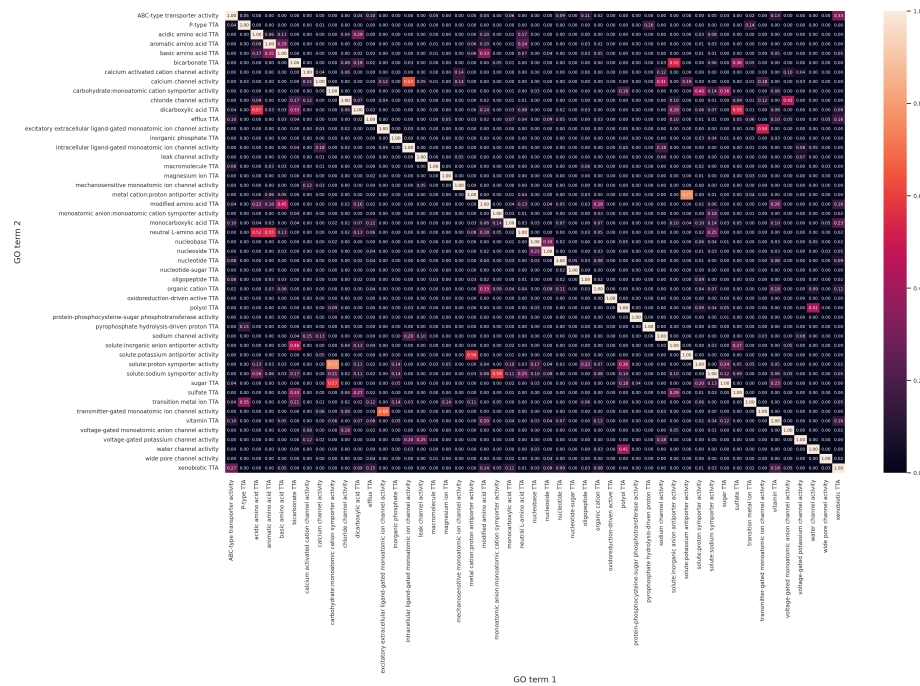


Figure 4.6: Heatmap showing the fraction of proteins annotated with GO term 1 that are also annotated with GO term 2, for all pairs of GO terms in the optimized subset for the meta-organism dataset.

setting m to 20 represents a trade-off between coverage and subset size. After the lowest possible redundancy is reached (as in Figure 4.6), the number of GO terms can be further reduced by lowering the coverage threshold until pairwise F1 scores are available for all GO terms at $m = 20$. This also has a positive impact on the pairwise F1 scores.

4.4.3 Comparison of meta-organism results with substrate classes used in previous studies

Finally, we compared the results of this study to previous machine learning works that classified a set of transmembrane transporters into substrate classes. For this, we searched the meta-organism dataset for GO terms that matched the corresponding substrate transport functions. We evaluated the resulting sets of GO annotations in terms of protein coverage, pairwise F1 scores, and annotation overlap. Previous studies used keyword annotations from UniprotKB to retrieve substrate classes, and one of these substrates was *electron*. However, in the Gene Ontology, which was used in this study as basis of functional annotations, *electron transfer activity* is classified as a molecular function that is separate from *transmembrane transporter activity*, rather than a descendant of it. Therefore, membrane proteins that enable reactions of electron donors and electron acceptors were not part of the new dataset, and were not available for comparison.

In 2010, Schaadt and Helms only considered four substrate classes of *phosphate*, *amino acid*, *oligopeptide*, and *hexose* [39]. In our dataset, using that subset of proteins would have only covered 17.8% of transporters. The overlap between the classes was small, only *amino acid* and *oligopeptide* had an overlap of 6%. The average pairwise F1 score was 0.83 ± 0.14 .

In 2014, Mishra et al. [40] also considered a category of *other transporters* based on data from version 2013.03 of Uniprot. However, it was unclear to us which substrates were included in that category. Furthermore, electron transfer proteins were not part of our dataset. The remaining five classes of *sugar*, *protein*, *cation*, *anion* and *amino acid* provided a protein coverage of 66.6%, with average F1 scores of 0.85 ± 0.13 . Three substrate classes (amino acid, anion and sugar) had overlaps of 22-36% with the cation class, in part because cation symporters and antiporters were included in the dataset. Otherwise, the overlaps were small.

Finally, we analyzed the substrate classes that we used in our previous study [349]. When excluding the electron transfer proteins, the remaining three substrate classes *sugar*, *amino acid* and *potassium* had no pairwise overlaps and covered 26.2% of the meta-organism dataset, with average pairwise F1 scores of 0.9 ± 0.05 .

By comparison, the meta-organism GO subset of 49 GO terms that was optimized for high protein coverage included 96% of transporters, with median F1 scores of 0.92 ± 0.1 . The median overlap was 0.0, with a small number of outliers that were necessary to reach the high coverage (see Figure 4.6). When accepting a lower coverage of 67%, we found a subset of 30 GO terms with median F1 scores of 0.95 ± 0.06 , and much smaller overlap between the classes

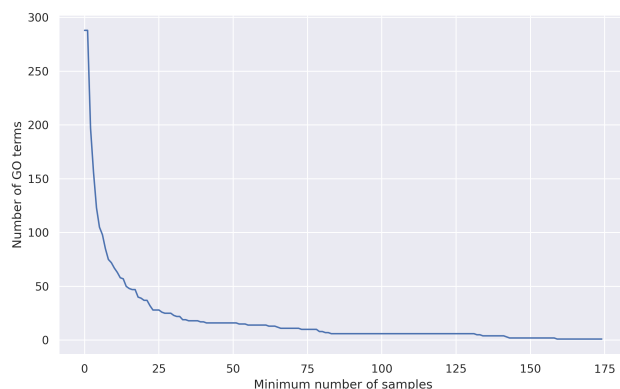


Figure 4.7: Number of GO terms available for training, at different thresholds for the minimum number of proteins associated with any GO term in the yeast dataset.

(see Figure C.5), meaning that the increased detail of the larger subset led to better classification performance and more consistent ML models.

4.4.4 Yeast transporter dataset

4.4.4.1 Results of the dataset creation pipeline

In a second stage, we analyzed a single-organism dataset, containing transporters from *S. cerevisiae*. Here, all transporters in our dataset are part of the manually curated SwissProt database, meaning that considering data from TrEMBL would not have increased the sample count. The gene names of all proteins in the dataset were known. The final protein dataset contained 332 transport proteins. Sequence clustering with CD-HIT [317] at a similarity threshold of 70% would reduce that number to 303.

When applying the annotation pipeline described in Section 4.3.2, the transporters were directly annotated with 211 unique transport-related molecular function GO terms. Adding the ancestors of these GO-MF terms to the dataset increased that number to 288. Overall, our yeast dataset contained a total of 3,299 unique relations between transport proteins and transport-related functional GO terms, 380 of which were inferred by electronic annotation (IEA).

4.4.4.2 Evaluation score matrix from machine learning models

Previously [349], we showed that at least 20 proteins per class are necessary to predict substrates of transmembrane transporters with good accuracy. Therefore, we pruned the GO terms in our dataset to those with at least 20 associated proteins. This represents a tradeoff, because the number of GO terms available for classification decreases rapidly if the minimum required number of proteins per GO term is raised (see Figure 4.7). Filtering the GO terms in this way did not decrease the number of proteins in our dataset, since they were still annotated with the ancestors of these more specific terms.

Next, we generated a sparse matrix containing the evaluation scores of pairwise machine learning classifiers, as described in Section 4.3.3.5. Each matrix entry is equal to the score of a multi-output SVM-RBF classifier that annotates a protein sequence with either one or both of the GO terms. Models were only trained on pairs of GO terms where each term was associated with m unique proteins, i.e. those that were not in the intersection set between the protein sets of the two GO terms. The final matrix had 37 dimensions. With $m = 20$, 524 out of 1,332 distinct pairs of GO terms were not suitable for evaluation due to a lack of unique samples in at least one of the classes.

After performing a nested cross validation for each pair of GO terms, the median F1 score on the independent test sets was 0.91 (see Table C.3). In order to test the impact of feature selection and dimensionality reduction on the scores, we trained two additional ML pipelines that employed either ANOVA-based feature selection or PCA to reduce the number of feature dimensions, based on the respective training dataset at hand. While feature selection did not substantially impact the median F1 score, it improved the minimum scores between pairs of GO terms from 0.15 to 0.23 and 0.37 for PCA and ANOVA, respectively. Overall, the model with ANOVA-based feature selection seemed to perform best.

Finally, we repeated the same tests after removing redundant sequences with more than 70% sequence similarity from the dataset using cd-hit. This reduced the number of proteins in the dataset from 332 to 303, and the number of GO terms with more than 20 associated proteins from 37 to 32. Median F1 scores on independent test sets between all GO term pairs were between 0.02 and 0.04 lower across the different models (see Table C.3).

4.4.4.3 Comparative analysis

Now we wanted to find out which pairs of GO terms can be well distinguished and which ones not. Intuitively, one may expect that it is more difficult to distinguish transporters of substrates A and B when either the protein sequences are highly similar to each other or when A and B are very similar. To this aim, we compared the model evaluation scores of binary machine learning models trained on pairs of GO terms to similarity scores for those GO terms.

The data was split into two subsets: GO term pairs with a macro-averaged F1 score below 0.75, and GO term pairs with an F1 score above or equal to 0.75. This threshold was selected based on the minimum sample size in each set, where a lower value would result in fewer than 30 GO term pairs in the subset of low-performing pairs, which could lead to biased results.

We tested using the Shapiro-Wilk test [327][328] which of the similarity scores follow a normal distribution. The median and mean sequence identities, and the GO semantic similarity had a test statistic higher than 0.9, indicating they follow a normal distribution.

For each of these normally distributed variables, we performed a one-tailed independent t-test between the two subsets of low-performing and high-performing GO term pairs, with the alternative hypothesis that the mean similarity score of the pairs with low F1 score was smaller than the mean of the population. Af-

Similarity measure	Spearman correlation	
	Mean F1 training set	Mean F1 test set
GO max sequence identity	-0.602	-0.555
Annotation overlap	-0.475	-0.421
Sample count diff.	-0.446	-0.420
GO semantic similarity	-0.331	-0.312
GO median sequence identity	-0.054	-0.032
GO mean sequence identity	-0.039	-0.017
GO min sequence identity	0.442	0.405
Mean F1 training set	1.000	0.957
Mean F1 test set	0.957	1.000

Table 4.1: Spearman correlations between F1 scores and similarity measures for the yeast dataset. Classification performance of ML models was calculated as the average F1 score. Correlation analysis was carried out to see if any similarity score can predict classification performance.

ter multiple-hypothesis correction by the Benjamini-Hochberg method, only the mean sequence identity ($p=0.0306$) was statistically significant with $p < 0.05$. No statistically significant similarity scores were found for the other tail of the distribution.

For the similarity scores that did not follow a normal distribution, we performed two non-parametric Mann-Whitney-U tests between the two subsets for each score, one for each tail of the distribution. After Benjamini-Hochberg correction, the mean sequence identity and the minimal sequence identity had statistically significant p-values, with the alternative hypothesis that pairs with lower F1 scores have a lower similarity score. For the other alternative hypothesis, i.e. that pairs with lower F1 score have a higher similarity score, we found the absolute difference in sample count, the maximum sequence identity between proteins annotated with the GO terms, and the semantic similarity to be statistically significant.

In summary, good ML separability between a pair of GO terms requires (unexpectedly) that the sequence similarity between the associated proteins should be higher than a certain minimum, but also (expectedly) should not exceed a certain maximum. The Wang method [384] for calculating semantic similarity between GO terms was also able to predict good ML performance, meaning that proteins annotated with more dissimilar molecular functions are also better separable by the ML model.

In addition, we also calculated correlation coefficients between F1 scores and similarity measures of all pairs in the dataset. Spearman's rank correlation was used, since most of the tested variables were not normally distributed. The F1 scores on the training and test data showed a very high correlation of 0.956,

GO term	Proteins
amide transmembrane transporter activity	21
carbohydrate derivative tta	25
carboxylic acid tta	64
monoatomic cation tta	131
organophosphate ester tta	22
passive tta	32
primary active tta	61
protein tta	21
salt tta	72
secondary active tta	80
sulfur compound tta	21

Table 4.2: Subset of GO terms selected for the yeast dataset when optimizing for protein coverage and prioritizing more specific terms. More general terms related to the mechanism are needed to increase coverage of the resulting multi-output model. *tta* = transmembrane transporter activity.

meaning that the trained models adapt well to the independent test sets. The highest correlation (0.44) was observed for the minimum sequence identity, the lowest correlation (-0.55) for the maximum sequence identity (see Table 4.1), which corresponds to the results obtained from the statistical tests.

Finally, we calculated mean F1 scores of individual GO terms across the pairs and compared them to a variety of attributes, including the number of annotated proteins and the level (i.e. the distance from the root node). The highest positive correlation with the test F1 score was found with the level (0.39), while the lowest (-0.61) was with the number of annotated proteins. This means that class labels should not be too abstract, since the set of protein sequences annotated with very broad GO terms is more diverse, making it harder for the machine learning model to find patterns.

4.4.4.4 Applying the GO term clustering algorithm to the yeast dataset

After removing all GO terms with fewer than $n = 20$ samples from the *S. cerevisiae* dataset, we were left with 36 out of 287 transmembrane transport-related GO terms.

Since the multi-output classifiers we used are capable of annotating proteins with one or both of the GO terms, we needed to filter out pairs where the minimum number of unique proteins per class, i.e. proteins that were not part of the intersection of the classes, was higher than a threshold m . We tested four different values for the parameter m : 5, 10, 15 and 20. Setting m to a value

GO term	Proteins
amide tta	21
carbohydrate derivative tta	25
inorganic molecular entity tta	158
macromolecule tta	21
monoatomic ion tta	142
organic acid tta	65
organophosphate ester tta	22
primary active tta	61
salt tta	72
secondary active tta	80
sulfur compound tta	21

Table 4.3: Subset of GO terms selected on the yeast dataset when optimizing for protein coverage prioritizing abstract terms. More general terms related to the mechanism are needed to increase coverage of the resulting multi-output model. *tta* = transmembrane transporter activity.

lower than five was technically not possible in our workflow, since we need at least five unique proteins for the 5-fold cross validation. Increasing m leads to a smaller number of pairs with F1 scores available in the pairwise evaluation matrix at high coverage values (see Figure C.6), but leads to more consistent median F1 scores (see Figure C.7). Based on these findings, we selected $m = 10$ as a trade-off between high coverage of the resulting subset and large enough sample count during evaluation of the pairwise SVM models. Before optimizing the subset, evaluation scores were available for 244 out of 1296 possible pairs, and the median F1 score was 0.90. Applying the greedy optimization algorithm described in Section 4.3.4 with $m = 10$, $\epsilon = 0$ and a preference for less abstract terms yielded a subset of 11 GO terms that cover 98.1% of all transmembrane transport related proteins in *S. cerevisiae*. F1 scores were available for all pairs of GO terms in the optimized subset, the median score was 0.87 (see Table 4.2). The pairwise overlap matrix of the subset (see Figure C.8) has a relatively low median value of 0.07 (arithmetic mean of 0.12), with a maximum pairwise overlap of 0.66. This is mostly due to the fact that abstract terms such as *primary active transmembrane transporter activity* need to remain in the dataset in order to reach the 98% coverage, since many of their child terms do not have enough annotated proteins to be included. For the meta-organism dataset, this issue was overcome when we included proteins from other organisms. Asking the pipeline to prefer more abstract terms did not have an effect on the subset that was found by the algorithm, with the given parameters.

Next, we tried other values of ϵ , along with ten different random seeds for the random number generator. Higher values of ϵ cause more GO terms to be

considered to have the same impact on the average F1 score, therefore making the results more dependent on the random draws that happen at the end of the pipeline. This can be used as an additional optimization step, in cases where the greedy algorithm finds a local minimum. Results were filtered for those subsets with high coverage (98% or more) and with F1 scores available for each pair. With $m = 10$, all resulting subsets were made up of 11 GO terms and a minimal variance between the results obtained with different random seeds and ϵ values. When telling the algorithm to prefer abstract terms and setting ϵ to 0.005, a subset with a slightly higher coverage was found (99% compared to the previous 98.1%), with a median F1 score of 0.87 (see Table 4.3). The resulting subset showed a higher average and median overlap (0.16 and 0.1, respectively), and a much larger maximum overlap of 0.91. While preferring abstract terms can increase the protein coverage, for example by selecting *organic acid* (65 proteins) instead of its child term *carboxylic acid* (64 proteins), the more generalized terms are more likely to annotate other proteins in the dataset, leading to larger pairwise overlaps.

4.4.4.5 Chemical similarity based on molecular fingerprints

Finally, we explored whether substrate ChEBI terms associated with GO terms (see Section 4.3.3.4) can be used to cluster a training dataset of transport proteins into substrate classes, by calculating a chemical similarity score for each pair of transporters in the training dataset, based on their substrates. This is done by retrieving a string representation for each molecule (called SMILES), transforming that into a binary vector using one of several algorithms, and then comparing these binary fingerprints with the Tanimoto coefficient (see Section 4.3.3.4). ChEBI annotation could also be useful to automatically filter out abstract transporter function GO terms, such as those that describe the type of mechanism, as these terms are not annotated with a molecule.

Of the 288 GO terms in the yeast transporter dataset, 221 were annotated with ChEBI terms, and 149 of those had SMILES representations. Only 15 GO terms had both SMILES of their primary substrates available, and were also annotated with 20 or more proteins from *S. cerevisiae*, which is a lower bound on the number of samples per class when training ML models to find transporter substrates [349]. These 15 GO terms annotate 173 of 332 transmembrane transport proteins in *S. cerevisiae*, meaning that only using GO terms that are annotated with ChEBI terms would lower the number of proteins and substrate classes in our dataset considerably. The main issue is that abstract GO terms are often linked to abstract ChEBI terms that are, in turn, not directly linked to SMILES. For example, the GO term *glucose transmembrane transporter activity* (GO:0005355) is linked to the ChEBI term *glucose* (CHEBI:17234), which does not have a SMILES string available. This is the case because *glucose* is a parent term in the ChEBI hierarchy that has a sub-tree of different molecules below it, such as D-glucopyranose, D-glucofuranose, aldehydo-D-glucose, and their L-glucose enantiomers. SMILES are typically only available for these leaf nodes in the ontology graph, which are, in turn, linked to fewer GO terms and proteins.

One idea was to combine the child molecules of an abstract term, such as *glucose*, into one SMARTS string. As an extension to SMILES, these provide additional

syntax that allows for wildcards, logical operators, rings or even recursion in the molecule representation, meaning that a set of similar molecules can be encoded with just one string. Algorithms for calculating the similarity between SMARTS have been developed [395]. However, this approach did not yield good results for our use-case. For example, the set of glucose molecules contains glucopyranose and glucofuranose in their closed- and open-chain conformations, both as D and L enantiomers, and with or without ^{13}C carbon isotopes. On the other hand, an eukaryotic glucose uniporter such as GLUT3 (Section 2.1.3.2) typically only transports D-glucopyranose, which means that combining all glucose molecules into a SMARTS string would not be a good imprint of its binding pocket.

Automatically finding a representative molecule for each abstract substrate term proved challenging, as a term like *hexose* can have a sub-graph of dozens or hundreds of molecules beneath it that can also contain cycles. This means that an appropriate SMILES string would have to be manually selected for each of the 288 transporter GO terms in yeast (679 GO terms for the meta-organism dataset) based on literature that tells us which sub-type of an abstract molecule description is actually present in a particular organism. This would contradict the premise of an automated annotation pipeline.

4.5 Discussion

Here, we aimed at identifying an optimal set of molecular function GO terms to split the members of a membrane transporter dataset, where optimality is meant in terms of separability. As an analogy, one could try to classify humans into those that like pop music, jazz, classical music or none of all. One question would then be whether it is beneficial to subdivide classical music into sub-categories, or to remain at the more abstract level of classical music. The analogy of music styles are GO terms at different abstraction levels of the Gene Ontology.

First, we implemented a software pipeline that automatically generates datasets of specified transmembrane transporter proteins, where the proteins and their functional annotations can be filtered for a variety of attributes, such as data quality, organism or the cellular compartment where they carry out their functions. Using this pipeline we created two transmembrane transporter datasets: A single-organism dataset containing only proteins of *S. cerevisiae*, as well as a meta-organism combining the transporters of human, *S. cerevisiae*, *E. coli* and *A. thaliana*.

Then we implemented an algorithm for reducing the redundancy in a set of GO terms, in order to find an optimal set of classes for training a machine learning algorithm. This greedy algorithm aims to minimize the pairwise overlaps of annotated protein sets, while producing subsets containing either abstract terms or specific terms. A filter for minimum and maximum protein count of the GO terms is employed, in order to reduce major differences in sample size.

When applying the pipeline to the meta-organism transporter dataset, the optimized subset contained 49 out of 695 functional annotations, which provided a coverage of 96%, a median pairwise F1 score of 0.92 ± 0.10 , and a mean overlap of 2%. Lowering the minimum required protein coverage to 67% reduced the subset size to 30 GO terms, with a median F1 score of 0.95 ± 0.06 and a mean

annotation overlap of 1%. This was an improvement on previous attempts at dividing transporters into substrate classes, which either provided a much smaller protein coverage, or led to lower pairwise F1 scores because of broader functional annotations. For the yeast dataset, the algorithm lowered the number of transporter GO terms from 287 to 11, with a median pairwise F1 score of 0.87 ± 0.16 and covering 98.1% of transmembrane transporters.

Also, we tested for the yeast dataset what characteristics of two sample groups determine whether they are well separable or not. To this end, we computed multiple pairwise similarity measures for the GO terms. On each pair of GO terms, we trained and evaluated a multi-output SVM model with RBF-kernel, using ML-features derived from the amino acid sequence and from evolutionary information. Model evaluation was carried out in a nested 5-fold cross validation approach, and the average F1 score across the five test sets was used as a metric for how well the model can distinguish proteins of these two classes. When only evaluating pairs of GO terms where each GO term has at least $m = 20$ unique samples available for training, the SVM-RBF model with our combined ML-feature and ANOVA-based feature selection achieved a mean F1 score of 0.88 ± 0.11 on the test sets for the *S. cerevisiae* dataset (see Figure C.3).

Then, we compared the similarity scores to these F1 scores on the yeast dataset using correlation coefficients and statistical tests. We found that the strongest predictor of low F1 scores was a low mean sequence identity between proteins annotated to the two different GO terms. On the other hand, both a lower minimum sequence identity and a higher maximum sequence identity between pairs of proteins annotated with the GO terms are predictors of low F1 scores. Together, this means that the protein sets should not be too similar, but also not too dissimilar to each other. Another predictor of a low evaluation score was a large difference in sample count, so comparing very abstract to very specific functional annotations can lead to poor results. The GO semantic similarity was also able to predict good separability with machine learning models.

While correlations with machine learning performance were observed, such as with minimum and maximum sequence identity (see Table 4.1), these correlations were not strong enough to make decisions about which annotations to keep and remove. Hence, we implemented a greedy algorithm that optimizes class selection based on the training dataset. Although the requirement of at least 20 samples per functional annotation prevents the inclusion of very specific GO terms, we addressed this limitation by adding the ability to create meta-organism training datasets that can include proteins from plants, bacteria, single-cell eukaryotes, and mammals. Multi-output machine learning algorithms, which allow training on proteins with multiple substrates are helpful in increasing the number of available samples per annotation.

Our method offers an automated solution to identify which functional annotations of transmembrane transporters can be separated with sufficient accuracy through machine learning. It provides a pipeline that vastly reduces the redundancy in a set of functional annotations, using a greedy algorithm and training data. Another pipeline allows for the creation of custom training datasets for a variety of organisms, including plants, single-cell eukaryotes, bacteria and mammals. This approach reduces the need for extensive manual effort in simplifying datasets, and helps in improving machine learning model efficiency, classification

performance, and training time. This brings us closer to the goal of creating predictive models that can automatically annotate entire proteomes and metaproteomes with transmembrane transport annotations and transported substrates, providing valuable information for various fields of research [16][14][6][8].

Chapter 5

Application of Protein Structure Encodings and Sequence Embeddings for Transporter Substrate Prediction

This chapter was published in the MDPI Molecules journal, on August 01, 2025 [396].

Code and raw data are available at https://github.com/adenger/subpred_dl.

5.1 Abstract

Membrane transporters play a crucial role in any cell. Identifying the substrates they translocate across membranes is important for many fields of research, such as metabolomics, pharmacology, and biotechnology. In this study, we leverage recent advances in deep learning, such as amino acid sequence embeddings with protein language models (pLMs), highly accurate 3D structure predictions with AlphaFold 2, and structure-encoding 3Di sequences from FoldSeek, for predicting substrates of membrane transporters. We test new deep learning features derived from both sequence and structure, and compare them to the previously best-performing protein encodings, which were made up of amino acid k-mer frequencies and evolutionary information from PSSMs. Furthermore, we compare the performance of these features either using a previously developed SVM model, or with a regularized feedforward neural network (FNN). When evaluating these models on sugar and amino acid carriers in *A. thaliana*, as well as on three types of ion channels in human, we found that both the DL-based features and the FNN model led to a better and more consistent classification performance compared to previous methods. Direct encodings of 3D structures with

Foldseek, as well as structural embeddings with ProstT5, matched the performance of state-of-the-art amino acid sequence embeddings calculated with the ProtT5-XL model when used as input for the FNN classifier.

5.2 Introduction

Due to advances in high-throughput sequencing and proteomics, the amount of biological data is growing exponentially. This enables a better understanding of biological complexity by revealing new protein families with common evolutionary origins, as well as shared three-dimensional folds derived from predicted protein structures. At the same time, it also enhances the possibilities for encoding this data, in order to sharpen the accuracy of predictive machine learning models. These models can, in turn, help us understand the biological functions that these genes and gene products serve in the living organism.

5.2.1 Motivation and Background

Membrane transporters are a crucial part of any cell, selectively transporting nutrients, metabolites, signaling molecules, and charged ions across its membranes, as well as drugs and xenobiotics. Determining the transport proteins that are present in a proteome, along with their transport mechanisms and their transported substrates, is therefore an important task that leads towards a deeper understanding of the connections between intra- and inter-cellular networks. Discovery of transporters for a particular substrate has implications for various fields, such as pharmacology and metabolomics. In biotechnology, the efflux of intermediary molecules is a common problem when optimizing micro-organisms for the production of a particular product.

After analyzing predicted 3D structures of human solute carriers, along with their homologues in other species, Ferrada and Superti-Furga [397] suggested that the 455 proteins share a total of 180 evolutionary origins. Furthermore, most of these proteins can be assigned to one of a small number of different structural folds [28][159]. The substrate specificity, however, does not necessarily depend on the fold and evolutionary origin of the protein, as it can evolve later throughout evolution. Human sugar transporters, for example, include the GLUT family of uniporters [167], which possess the MFS fold [29], while the SGLT family of sugar-cation symporters displays the LeuT fold, making it part of the APC superfamily, which is mainly made up of amino acid transporter families [177]. In 2010 [398], the SWEET family of bi-directional sugar uniporters was found in *A. thaliana*, and homologues have been discovered in prokaryotes and animals, including one human SWEET transporter [399]. Prokaryotes also possess ABC transport complexes with unique folds that import nutrients, such as amino acids and sugars [34].

5.2.2 Membrane Transporter Substrate Prediction with Machine Learning

Due to this complexity, one cannot solely rely on similarity of sequence and structure to tell transport proteins apart by their substrates, and there is a

need for developing specialized protein encodings that can capture the necessary information, beyond evolutionary relationships and shared protein domains. Machine learning methods trained on protein sequence have successfully been used for this purpose. Schaadt and Helms [39] encoded protein sequences through amino acid frequencies, as well as through data from multiple sequence alignments (MSAs) [366], to assign transporters in *A. thaliana* to four substrate classes. Other approaches incorporated evolutionary information from position-specific scoring matrices (PSSMs) [370], as well as physicochemical properties of residues in the sequence [40][367], to classify transporters in manually assembled subsets of Swiss-Prot. More recently, deep learning architectures have also been applied in this field. For example, the tool SPOT used transformer networks to predict whether an arbitrary chemical entity is likely to be a substrate of a given transporter or not [400]. The tool NA.cNN tested four current protein language model embeddings of amino acid sequences, and was able to identify sodium transporters among all membrane proteins with high accuracy [401].

In a previous study [349], we compared and combined existing protein sequence features into a META-feature, and tested their performance across different machine learning models and transporter datasets. We then compared model performance to various similarity measures between substrate classes and protein annotations through correlation analysis and statistical testing, and introduced an optimization algorithm that identifies a set of substrate classes for a dataset of transmembrane transporters with minimal redundancy and high pairwise classification performance [376]. As a basis for that work, we also developed a dataset pipeline for creating annotated protein sets, where constraints on predicted and experimental data allowed for a trade-off between the quantity and quality of the training data.

5.2.3 Applications of Deep Learning Models in Bioinformatics

Recent advances in deep learning, such as transformer models [402], convolutional neural networks (CNNs) [403], graph neural networks (GNNs) [404], and variational autoencoders (VAEs) [405], have successfully been applied to a wide range of tasks in computational biology. This includes feature extraction from protein sequences [378][406], protein sequence generation from text [407][408][409], as well as various predictive tasks, e.g., for secondary and tertiary structure [21][410][411], structural similarity and protein family membership [412][413], GO term annotations [414][415][416][417][418], protein-protein interactions (PPIs) [419][420], sequence variant effects [421][422], sub-cellular location [423][424], and many more.

Transformer-based protein language models (pLMs), such as ProtT5 [378] and ESM [406], have been trained on proteomics datasets containing a very large number of sequences [418]. These pre-trained models can now be used for feature extraction from amino acid sequences in the form of language embeddings, or they can be fine-tuned, e.g., for protein sequence generation [425]. To apply these methods from natural language processing (NLP) to proteins, each amino acid in the sequence is treated as a word, and encoded by a numerical token. The encoder of the transformer model then converts these words into fixed-length numerical vectors. These vectors can be averaged (mean-pooled) across

the sequence to create a vector representation for the entire protein. Both the per-protein and per-residue embeddings are suitable as input for other models, to predict properties of the entire protein or of individual sequence positions, respectively, through supervised learning [417][424][416][425][418]. As similar proteins produce similar embeddings, they can also be used for unsupervised learning techniques such as clustering, and for visualization [426]. In addition to embeddings, another commonly found method for encoding protein sequences is that of trigrams [427][428], which we refer to as 3-mer frequencies. Previous approaches for membrane transporter substrate prediction trained models on k -mer frequencies with $k \in \{1, 2\}$ in the form of amino acid composition and pair amino acid composition (AAC and PAAC, respectively). Yet, it would be interesting to compare the performance of $k = 3$.

5.2.4 Predictive Methods Based on Protein Structures

Due to the high accuracy of protein 3D structural models predicted from sequences with AlphaFold [21] and similar models [411][410], and the resulting availability of protein structures for hundreds of millions of proteins [275], a new generation of bioinformatics software has emerged that uses this data for data mining and prediction, often along with features derived from the amino acid sequence, from evolutionary information in the form of PSSMs or MSAs, and/or from known protein-protein interactions (PPIs) [429] [351] [430] [431]. Some methods, such as DeepFRI [351], GAT-GO [431], GNNGO3D [429], and Panda3D [430], encode the structures directly, typically by calculating contact maps or distance maps between α -carbon atoms in the amino acids. Another technique for encoding protein structures, which is already used by some homology-based predictive models [432][42], exists in the form of 3Di sequences.

These 3Di sequences were introduced together with the 3D structure search tool Foldseek [42], and aim to encode a protein structure as a structural sequence that has the same length as the respective amino acid sequence, and the same alphabet size of 20. The 3Di state of a residue takes its nearest neighbor into account, along with surrounding positions in the sequence. Based on descriptors derived from these residues and their relative positions in the structure and sequence, a VAE model finds 20 discrete states with high conservation among the protein structures in its training dataset. The 3Di sequence of a protein can then be used for sequence alignments with the help of a substitution matrix to compare the folds of two proteins, or for fast database searches to find similar structures, and potentially infer functions or properties through structural homology [42][432].

So far, 3Di sequences have only been used as ML training data by a few predictive models, e.g., the Topsy-Turvy 3D (TT3D) model [433] for predicting PPIs, which encoded 3Di sequences by one-hot encoding, and combined that with amino acid embeddings from a pre-trained pLM. Furthermore, the ProstT5 model [434] is a Text-to-Text Transfer Transformer (T5) [435] that aims to translate directly between amino acid and 3Di sequences, and was trained on both. As it is a transformer model, its encoder can be used to create embeddings for either type of sequence.

5.2.5 Outline

The goal of this study is to test whether recent advances in deep learning lead to better models for membrane transporter substrate prediction. To this end, we compared previously published methods, which train SVMs on amino acid k-mer frequencies and evolutionary information from PSSMs, against features generated with various deep learning models, such as amino acid embeddings from pLMs, as well as features derived from 3Di sequences. The structural models for calculating the 3Di sequences were, in turn, predicted from the protein sequence with AlphaFold. We tested the performance of 3Di embeddings with ProstT5, and used 3Di k-mer frequencies as ML-features for prediction, both of which, to our knowledge, have not been tested so far. Finally, we tested the impact of replacing our previous SVM model with a regularized feedforward neural network (FNN).

5.3 Methods

5.3.1 Data Availability

All code used in this study is provided in a git repository at https://github.com/adenger/subpred_dl (accessed on 28 July 2025), together with a reproducible coding environment that contains all dependencies. Links to archives containing raw and pre-processed data can be found in the description of the repository. Together with the code and environment, the results found in this study can be fully recreated. The “preprocessing” folder contains all scripts that were used to download and process the raw data.

5.3.2 Protein Dataset Creation

Annotated protein datasets were automatically assembled with a dataset creation pipeline. Protein sequence data was retrieved programmatically from the Uniprot [245] API (version 2025_02). This table contains all proteins from Swiss-Prot (reviewed) and TrEMBL (unreviewed), along with accessions from other databases, such as Uniprot Keywords, Interpro [294], gene names, and Transporter Classification Database (TCDB) [233] identifiers. Only proteins with non-fragmented sequences were retrieved. We only kept proteins in the dataset where the existence of the protein sequence had been experimentally verified at the protein level, and where the encoding gene was known.

As some of the machine learning features used in this study were calculated on predicted 3D structures from AlphaFoldDB [21, 275], which does not consider proteins with non-standard amino acid codes such as B, O, U, Z, and X, we removed these proteins from our datasets.

The code for the dataset creation pipeline was adapted from our previous studies on transmembrane transporter substrate prediction [349, 376], and can now create datasets for any task that involves the prediction of GO terms for sets of protein sequences or 3D structures, not just transporter datasets.

Sequence clustering was carried out on the dataset with CD-HIT [317] to prevent two very similar proteins from the same class being part of the training and test

sets. As in previous studies, we employed a threshold of 70% sequence identity [349].

5.3.3 Gene Ontology Graph and GO Annotations

Functional annotations about substrate specificity were retrieved as Gene Ontology [47] (GO) terms from the *molecular function* aspect of the Gene Ontology. The full dataset of Gene Ontology annotations [288] for Uniprot (GOA-Uniprot-All version 226) was downloaded from the FTP server of the European Bioinformatics Institute (EBI) in the GO annotation format (GAF, version 2.2), and subsequently filtered for relevant columns and rows using AWK (version 5.3.1) and Python (version 3.12.10). During filtering, only annotations of Uniprot accessions were kept, along with information about the evidence quality, and about the type of relation between each protein and GO term. In a final pre-processing step, all but the proteins from Section 5.3.2 were removed, reducing the compressed file size from 21 GB to 280 MB.

Relations between the GO terms were downloaded from the official GO website, in the OBO format [280] (database version 2025-03-16). They were read into a NetworkX (version 3.4.2) [436] graph data structure using the obonet Python package (version 1.1.1).

When a machine learning protein dataset with GO annotations was created with specified parameters, the implicit GO annotations for all proteins were derived from the explicitly listed annotations and added to the dataset, using graph search methods from the NetworkX package. When a protein was annotated with a specific GO term, and that GO term was connected to more abstract GO terms through the “is_a” relation, then the protein was annotated with those ancestor terms as well.

5.3.4 Blast Databases and PSSM Calculation

Position-specific scoring matrices (PSSMs) were calculated with PSI-BLAST [299] on local protein-BLAST databases created from Uniref50 and Uniref90 (version 2022_05) [267]. Both the PSSMs and the BLAST databases were generated with the blast+ package (version 2.16.0) [316]. We calculated PSSMs for each protein with either one or three iterations of PSI-BLAST on either of the two BLAST databases, with an E-value threshold of 0.002. PSSMs were stored in a cache folder, which greatly reduces the runtime of subsequent runs. A script for updating the BLAST database is provided in the repository.

5.3.5 Predicted 3D Structures and 3Di Sequences

Predicted 3D structures were downloaded from the AlphaFoldDB (version 4) [21, 275] FTP server for four model organisms: *A. thaliana*, *E. coli*, *S. cerevisiae*, and human.

The 3D structures of all proteins in the AlphaFoldDB archives were converted to 3Di sequences [42] using the Foldseek command line program (ver. 10.941cd33). To this end, a custom Foldseek database was created from all extracted PDB files, and subsequently converted to a Fasta file containing the 3Di representations.

Human proteins with a sequence length of more than 2700 amino acids were split into multiple, overlapping PDB files by AlphaFoldDB, where each subsequence (usually except for the last one) was 1400 amino acids long and had an overlap of 1200 positions with the next substructure. We created the 3Di sequences for each of these structure fragments separately, and then merged the 3Di sequences back into a single sequence by starting with the first sequence and subsequently adding the suffix after position 1200 from each of the following splits (see Section 5.4.1.1 for statistics).

5.3.6 Machine Learning Features Derived from Amino Acid Sequence

Amino acid sequences were encoded using three different feature generation algorithms that turn variable-length amino acid sequences into fixed-length numerical vectors.

The amino acid composition (AAC) is the frequency of each amino acid type in the standard genetic code, divided by the length of the sequence. The pair amino acid composition (PAAC) is the frequency of each possible pair of two amino acids, divided by the total number of pairs in the sequence (i.e., its length minus one). We also created a tri-peptide composition from the relative frequencies of all possible subsequences of length 3.

The PSSM feature was derived from a position-specific scoring matrix. For a sequence of length n , this PSSM is a $n \times 20$ integer matrix, containing the log-odds scores of exchanging the amino acid type at any position i ($1 \leq i \leq n$) with any of the 20 amino acid types throughout evolution.

For all sequence positions with a particular amino acid type, the log-odds vectors at those positions are added together along the sequence axis, and divided by the number of occurrences of that amino acid type in the sequence to calculate the average vector. The resulting 20×20 matrix reflects the evolutionary conservation profile of the protein.

Finally, the features were combined in different ways. As the PSSM matrices for each protein all contained log-odds values, we concatenated their rows into one Multi-PSSM feature that includes data from closely and distantly related proteins, due to different choices of BLAST databases and PSI-BLAST parameters for each feature matrix. In a previous study [349], we created the META-feature from AAC, PAAC, and multiple PSSMs by standardizing each feature along the sample axis and then concatenating the standardized vectors. This avoids information sharing as each sample is standardized independently of the others, but can change the scales along the feature dimensions, meaning that the META-feature should be seen as a fingerprint of the amino acid sequence.

As most of these features had more dimensions than we had samples in our dataset, we employed feature selection, dimensionality reduction and regularization techniques, along with a sound evaluation procedure, to prevent and test for overfitting.

5.3.7 Machine Learning Features Derived from 3Di Sequence

The 3Di sequences can be used as input for many methods that were originally invented for amino acid sequences, as similar structures lead to similar sequences (where the similarity can be calculated with a provided substitution matrix). Therefore, we applied AAC, PAAC, and KMER3 feature generation to these structural sequences as well, resulting in the “3Di_COMP”, “3Di_KMER2”, and “3Di_KMER3” features, respectively.

Finally, the information from amino acid and structural sequences were combined by concatenating AA and 3Di k-mer frequencies of the same lengths into the “COMB_KMER1”, “COMB_KMER2”, and “COMB_KMER3” features.

5.3.8 Embeddings of Protein Sequence and Structure

The proteins sequences were encoded with two different Text-To-Text Transfer Transformer (T5) protein language models (pLMs) [435]. Converting a sequence through the provided tokenizer and using it as input for the model resulted in a 1024-dimensional embedding for each token, i.e., a numerical vector of length 1024 for each position in the input sequence. These per-residue embeddings were then turned into per-protein representations with mean-pooling, where the average embedding for each type of token (e.g., each amino acid type) was calculated.

Pre-trained T5 models were retrieved through the Huggingface transformers API [437], and subsequently used to generate protein embeddings locally. The ProtT5 model [378] was used to encode amino acid sequences, resulting in a 1024-dimensional numerical feature vector for each protein in the dataset after mean-pooling. Furthermore, we employed the encoder of the bilingual ProstT5 model [434], which aims to translate between AA and 3Di sequences directly, to encode both AA and 3Di sequences. To this end, we used the corresponding T5 encoder, along with the provided tokenizer, to encode either 3Di sequences or AA sequences into mean-pooled per-protein embeddings.

5.3.9 Dummy Feature

We also added a “DUMMY” feature in each test, which consisted of 1024 positions that were deterministically drawn from a uniform random distribution with numpy [438]. The purpose of this non-informative feature was to estimate the amount of additional information that is provided by the real features calculated from the proteins, and how much more stable the resulting models are. A high score of the DUMMY feature can be a sign that the model is overfitting, as this can cause the model to find non-existent patterns in the data.

5.3.10 Support Vector Machine (SVM) Models

All tasks related to training and optimizing the SVM model pipeline were carried out with scikit-learn (version 1.6.1) [56]. The SVM models [340] used an RBF kernel. The corresponding regularization parameter γ , where smaller values reduce the influence of individual samples on the decision boundary, was

automatically scaled inversely to the number of features and the variance in the training data, as

$$\gamma = \frac{1}{\text{var}(X_{\text{train}}) \cdot m},$$

where m is the number of features and X_{train} is the training split of the feature dataset.

The value of C , which determines the penalty for samples located inside the decision boundary, was optimized between three values through hyperparameter optimization ($C \in \{0.1, 1, 10\}$). C was multiplied with different class weights for each class i , calculated as

$$\text{weight}_i = \frac{n}{K * n_i},$$

where n denotes the total number of samples, K is the number of classes, and n_i represents the number of samples annotated with class i .

Feature selection and pre-processing were carried out in three steps. To avoid information sharing, all constants and scores were derived from the training set and only applied to the test set. First, all features with a variance of exactly 0 in the training data were removed. This is especially important for the KMER3 features, as there are $20^3 = 8000$ possible 3-mers for an alphabet size of 20, many of which did not appear in any protein. For each of the remaining features, ANOVA F-scores were calculated based on the labeled training data, and the k features with the highest score (i.e., those that showed the highest variance between classes relative to the variance within the classes) were selected. The number k was optimized through hyperparameter optimization, and was limited to $k \leq \min(200, m')$, where m' is the number of feature dimensions left in the dataset after initially applying the variance threshold. Finally, the selected features were standardized by calculating z-scores along the feature axis, with mean and standard deviation derived from the training split.

The hyperparameter optimization was carried out through a grid search, where each combination of parameters was tested in a 5-fold stratified cross-validation on the training split. To obtain an even performance between both classes, the hyperparameters were optimized with the balanced accuracy score.

5.3.11 Deep Neural Network (DNN) Model

Deep neural networks (DNNs) for binary classification of substrates were created with TensorFlow (version 2.18.0) [439] using the Keras [440] Sequential API. The shape of the input layer was matched to the dimensions of each feature. This was followed by three densely connected hidden layers using the ReLU activation function and containing 512, 256, and 128 nodes, interspersed with dropout layers that aim to prevent overfitting and improve generalization by randomly cutting 50% of the connections during training. Finally, a single output node with a sigmoid activation function provided the predicted probability score $p \in [0, 1]$ that a given sample was a member of the positive class. During inference, a sample is assigned to the positive class if $p > 0.5$, otherwise it was predicted as negative.

The model was optimized with the Adaptive Moment Estimation (Adam) [441] optimizer, using binary cross-entropy as the loss function. To account for the low sample count in our datasets, we trained the model using a batch size of eight samples and limited the number of training epochs to 50.

5.3.12 Automatic Outlier Detection

After the feature generation was finished, the individual feature datasets were first submitted to an outlier detection pipeline. After standardization, each feature dataset was converted to a number of principal components that explain at least 95% of variance in the data, using principal component analysis (PCA). Next, an Isolation Forest [442] assigned an anomaly score to each sample and used that, together with an automatically determined anomaly threshold, to classify samples as outliers. If a sample was found to be an outlier in at least 50% of feature datasets, it was flagged as an outlier for further analysis and can optionally be removed. As this detection was performed using the entire dataset, we excluded this step when evaluating models with cross-validation.

5.3.13 Evaluation Methods

Evaluation of the SVM model was carried out in a nested, stratified 5-fold cross-validation (5FCV) with five repetitions. For the outer cross-validation, the dataset was split into five parts of similar size. Each of the five subsets was used once for validation, while the remaining four were combined into a training set. On this training set, internal 5FCV's were carried out with every combination of parameters, and the best set of parameters across all five splits was selected. The model was then fitted to the entire training dataset with the best parameters, and tested on the validation set. As the validation sets can be relatively small, an additional true positive can have a considerable impact on the score. To reduce the impact of this stepping on the results, we shuffled the dataset five times in a deterministic fashion and repeated the outer 5FCV for each of the shuffled datasets. The final evaluation score was then calculated from all $5 \times 5 = 25$ test scores from the evaluation sets. Stratified splits ensured that the ratio between the classes remained as close to the full dataset as possible.

As both classes had equal importance and some of the datasets showed a class imbalance, the hyperparameter optimization was carried out with the balanced accuracy score, i.e., the average recall score across all classes. To detect a class imbalance between the two substrate classes, an F1 score was calculated for each class individually. As the F1 score is the harmonic mean between precision and recall, a high F1 score on both classes means that both precision and recall for either substrate class are sufficient. Finally, we estimated overall performance through the balanced accuracy score and macro-averaged F1 scores. These metrics are calculated as the arithmetic mean between the performance on the individual classes, meaning that lower scores have to be interpreted more carefully.

The DNN model was evaluated on the same training and test sets that were used for the outer loop during the SVM evaluation, but did not require the inner loop as we did not adjust any hyperparameters based on training data.

Regularization was instead carried out by randomly deactivating connections during training.

5.4 Results

5.4.1 Classifying Substrates of Solute Carriers in *Arabidopsis thaliana*

5.4.1.1 Dataset Analysis

As a first task for the new machine learning features, we trained a model to distinguish between amino acid transporters and sugar transporters in *A. thaliana*. We used our dataset creation pipeline (see Sections 5.3.2 and 5.3.3) to generate a dataset of 33 amino acid and 32 sugar transporters, excluding those where protein data and GO annotations were not manually curated. After applying sequence clustering at a 70% threshold, the numbers of sugar and amino acid transporters were reduced to 28 and 26, respectively. The raw data for all ML-features was available for every transporter.

According to InterPro [294], 13 of these sugar transporters were members of the SWEET (Sugars Will Eventually be Exported) family (TCDB accession 2.A.123), 15 were part of the Major Facilitator Superfamily (MFS, TCDB 2.A.1), and one protein (SWEET4) was annotated with both. MFS and SWEET proteins are structurally distinct, and contain a different number of transmembrane helices (TMHs). SWEET4 contains seven TMHs arranged into two bundles of three TMHs with one connecting helix in the middle, as is typical for members of the SWEET family, whereas the MFS superfamily is defined by its characteristic twelve TMHs arranged into two bundles of six [29][399]. The nucleotide sugar transporter URG6 was the only protein in the dataset that was annotated with neither family. When looking at the names of the MFS proteins, we found that most transport hexoses and pentoses, most commonly as uniporters, but two xylose-proton symporters exist as well. In addition to the aforementioned substrates, there was one transporter each for sucrose, polyol, and ascorbate, which are grouped under sugar transmembrane transport by the Gene Ontology.

The amino acid transporters were not as thoroughly annotated with family names. The *amino acid transporter, transmembrane domain* (InterPro accession IPR013057) was found in 13 out of 26 proteins. The remainder included four members of the mitochondrial carrier domain superfamily, and four proteins with the *cationic amino acid transporter, C-terminal* domain. The three domain annotations were mutually exclusive. The remainder of the dataset was annotated with domains or families that occurred only once. None of the amino acid transporters in the dataset were annotated as MFS transporters.

When analyzing the dataset, we found that a glutamate-gated, monoatomic ion channel (AtGLR3.3) ended up in our dataset. This was caused by a faulty relation in GO, which listed glutamate-gated ion channels (GO:0004970) as a sub-type of amino acid transmembrane transporters (GO:0015171), contradicting the respective definitions of the terms. This connection was added in 2023, meaning that it did not influence our previous studies, as they used data from

2022. We kept the protein in the dataset to test the performance of training datasets created with our pipeline, and tested the impact of removing it when evaluating the automatic outlier detection method.

5.4.1.2 Feature Comparison

We calculated the features for all proteins in the dataset, and subsequently trained and evaluated SVM models in a nested 5-fold cross-validation approach. Due to the low sample count in the test set, we decided to shuffle the dataset five times, and repeated the evaluation procedure for each shuffle. The distribution of the scores was then calculated from all 5*5 scores on the respective test sets.

The hyperparameters of the pipeline, including the regularization parameter C of the SVM and the number k of features to select according to ANOVA F-scores, were optimized by calculating balanced accuracy on the training set. The maximum number of features that could be selected for each model was set to 200, in order to improve the generalization, and C was selected as one of three values ($C \in \{0.1, 1.10\}$). For the test set of each split, the F1 scores of the individual classes were calculated, in order to test for classification performance and class imbalance. The balanced accuracy and macro-averaged F1 score were also calculated as general performance measures across both classes.

The means and standard deviations for the features are shown in Table D.1, while Figure 5.1 visualizes their distribution. Since our dataset only contained 54 samples in total, a single false positive or false negative could have a large effect on the resulting score distribution. As there are only 10 or 11 samples in each test set, the accuracy score would increase in steps of 9–10%. We repeated the CV five times to obtain more accurate median scores, whereas the variance might not be representative of the actual performance.

The performance of the dummy feature was as expected for numbers drawn from a uniform random distribution, which confirms that the results are not caused by overfitting or similar issues, and that the remaining features provide additional information compared to randomly drawn values. The slightly higher F1 score for sugar transporters can be traced back to the higher class homogeneity, as all sugar transporters belong to two families, while the proteins in the amino acid class belong to a larger number of families, and therefore potentially have more than two evolutionary ancestors. Both classes had medium F1 scores of 0.5 or lower, and the balanced accuracy was 0.463.

Next, we tested the previously best-performing features for membrane transporter substrate prediction [349, 376]. The simple amino acid composition (AAC) with only 20 positions did not capture enough information, leading to a median score of around 0.71, with a large variance. In contrast, PAAC performed surprisingly well on this particular dataset, possibly due to the fact that most proteins in our dataset contain one of a small number of sequence domains that are associated with their respective substrate specificity. The amino acid 3-mer frequency (AA-KMER3) did not perform better than the 2-mer encoding (i.e., PAAC).

The Multi-PSSM feature led to models that performed reasonably well, with the majority of CV splits producing F1 scores between 0.7 and 0.9. Similarly

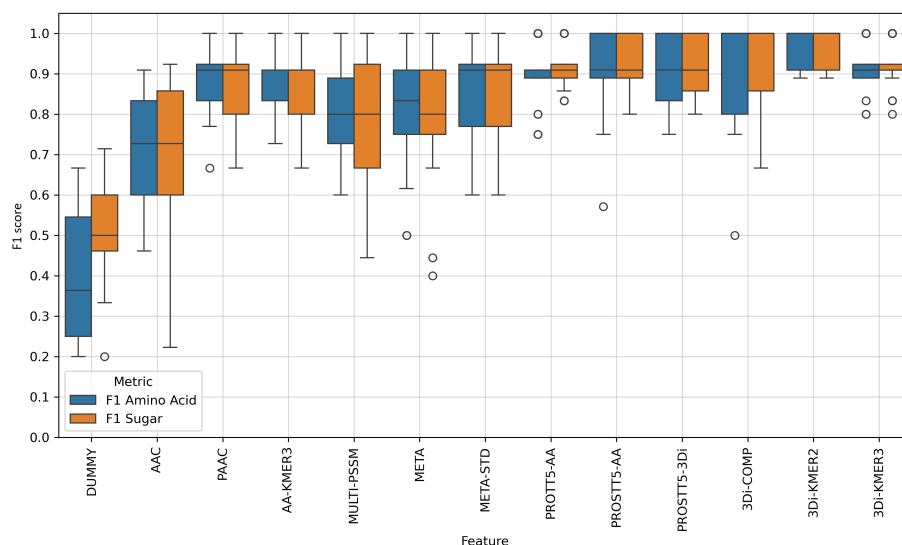


Figure 5.1: Evaluation results for an SVM model that distinguished sugar from amino acid transporters in *A. thaliana*. This plot visualizes the distribution of F1 scores for each class after five shuffled repetitions of a nested 5-fold cross-validation. Each box represents the 25th to 75th percentile, the line in the box is the median, and dots are test cases that were classified as outliers. A score below the median is determined to be an outlier if the distance between the 25th percentile and the score is greater than the distance between the 25th and 75th percentiles multiplied by 1.5. For scores above the median, the distance to the 75th percentile is used instead. Table D.3 provides a short overview of the features, in terms of the data that was used to calculate them. Due to the number of available samples for evaluation, the size of each test set was relatively small (10–11 samples), meaning that the median score calculated from the 25 splits is more representative of real-world performance than the variance. Both the embeddings and the features derived directly from the 3Di sequence outperformed previous approaches, which were based on evolutionary information and amino acid k-mer frequencies.

to the AAC, the variance on the sugar class was much larger than on the amino acid transporters. This effect was reduced to two outliers when combining the Multi-PSSM feature with AAC and PAAC into the META feature. The protein fingerprint feature META-STD, which is standardized along the sample axis, no longer produced outliers with F1 scores below 0.6, and had an improved median score (see Figure 5.1).

The variance in performance between CV splits was much lower for the amino acid embeddings created with ProtT5, whereby only one split led to an F1 score of 0.75 on the amino acid transporter class. A total of 20 out of 25 scores are located within a window of size 0.05 around the median score of 0.91. In addition to generating amino acid sequence embeddings with the ProtT5 model, we also created embeddings with the bi-lingual ProstT5 model. This encoder-decoder transformer model aims to translate directly between amino acid and 3Di sequences, although we only used it to encode either protein sequences from Uniprot or 3Di sequences generated from predicted protein structures, as protein sequence embeddings or protein structure embeddings, respectively. The score distributions between the two models only differed slightly, although there is one split where the amino acid embeddings did not classify the amino acid transporter correctly, while the 3Di embeddings did (see Figure 5.1).

Since we already calculated 3Di sequences for all protein sequences in our dataset with data from AlphaFoldDB, we also explored the idea of turning them into k-mer frequency features, similar to AAC and PAAC. These 3Di features worked surprisingly well, even beating the embeddings, both in their mean scores and in their consistency across the cross-validation folds. The 3Di 2-mer frequency “3Di-KMER2” worked especially well, capturing enough information about the proteins to reach a mean balanced accuracy of 0.967, with a standard deviation of only 0.046 across all 25 independent test sets, while also showing minimal class imbalance between the sugar and amino acid carriers. Simply comparing the frequencies of the 20 3Di states between the two protein classes was enough for a balanced accuracy of 0.93 ± 0.1 , which is impressive when considering that the resulting feature vector only consists of 20 positions. However, the distributions of F1 scores in Figure 5.1 suggest that the 2-mer frequencies lead to the most stable model, as there were some edge cases where the performance of 3Di-COMP dropped sharply, and where the 3Di-KMER3 produced outliers in the score distribution. All features were evaluated on the same splits, meaning that the 2-mer frequency managed to classify these cases correctly.

As 3Di k-mer frequencies seemed to work so well for encoding substrate binding affinities of membrane proteins, we tried combining them with the amino acid compositions to see if that additional information improved them further. The performance of these combined features (see COMB features in Table D.1) was comparable to the embeddings, while producing a higher score variance and lower mean values than the best-performing 3Di-KMER2 feature.

5.4.1.3 Automatic Outlier Removal

As mentioned before, our automated dataset creation methods sometimes identified a few outliers in the dataset. This can be caused by faulty connections in the GO graph as we found above, or by biological differences between one protein and the rest of the dataset. For example, the proteome of *E. coli*, a gram-negative bacterium that is surrounded by an inner and an outer cell membrane, includes outer-membrane pores (OMPs) that enable a wide range of nutrients and signaling molecules to enter the periplasmic space, including sugars and amino acids. Having these β -barrel pores in the same training dataset as α -helical solute carriers of those molecules can have a very detrimental effect on model training and classification performance.

To automatically handle cases like this, we implemented an automatic outlier detection method using PCA and Isolation Forests (see Section 5.3.12). This method found the aforementioned glutamate-gated ion channel AtGLR3.3 as an outlier, along with the chloroplast outer-envelope pore protein OEP161, and a lysosome membrane cystine transporter called “Cystinosin homolog”. When enabling the automatic outlier removal during the dataset creation process and repeating the evaluation procedure for our SVM model with all features (see Table D.2), we found that the 3Di-KMER2 feature still delivered the best-performing model, with similar mean and standard deviations to the evaluation scores, while the average balanced accuracy for the embeddings improved by 0.11 for the ProstT5-AA feature, and by 0.04 and 0.035 for ProstT5-3Di and ProstT5-AA, respectively. This means that the embeddings generated with the ProstT5 model performed better than the ProtT5 embeddings on the cleaner

dataset, while also delivering more consistent results across the 25 test sets (see Figure D.1). Another interesting observation is that the COMB-KMER1 feature, which combines the frequencies of 20 amino acid types and 20 3Di states into a feature vector of 40 positions, now led to better results than the 3Di-COMP feature alone, with a higher mean score and lower standard deviation, and produced a much better model than the amino acid composition by itself (see Table D.2).

Finally, it should be noted that the outlier removal pipeline was applied to the full dataset before the evaluation, meaning that the evaluation results from this particular test might not reflect the performance on previously unseen data. The purpose of this test was to obtain an estimate of how the model would perform on a perfectly clean dataset, and assess the impact of potential errors introduced by the automated dataset creation pipeline.

5.4.1.4 Deep Neural Network Models for Membrane Transporter Substrate Prediction

In a previous study [349], we showed that a single-layer perceptron can improve membrane transporter substrate prediction, compared to SVMs. This time, we also wanted to explore how a deep neural network, created with TensorFlow and the Keras API, would compare.

Our DNN model had three hidden layers, with 512, 256, and 128 nodes, respectively. The size of the input layer was adapted for each feature vector (e.g., 1024 for the embeddings, or 400 for 3Di-KMER2). To improve generalization and reduce overfitting, the hidden layers were interspersed with dropout layers that randomly (but deterministically) cut half of the connections during training.

The model was evaluated in a 5-fold stratified cross-validation approach, with five repetitions, where the data was randomly shuffled each time. The same datasets, random seeds, and cross-validation splits were used as for the SVM model, meaning that both models were evaluated on exactly the same training and test splits, without employing outlier removal.

The distributions of F1 scores in Figure 5.2 show a higher consistency across the individual feature datasets, with fewer outliers than the SVM results in Figure 5.1. This is especially the case for the embeddings and for the 3Di features. The META and PSSM scores delivered improved results, and the embeddings and 3Di compositions achieved high median scores. The best and most consistent results were obtained with the amino acid embeddings from the bi-lingual ProST5 model, and with the 3Di 2-mer frequencies, which was also the best-performing feature for the SVM model. For both of these features, the lowest F1 score on any class across all 25 independent test sets was 0.89. Interestingly, the results were more consistent than those of the SVM model after outlier removal (as shown in Figure D.1), reflecting the common observation that DNNs with multiple layers tend to be less sensitive to outliers than binary SVM models.

When comparing the means and standard deviations of the balanced accuracy scores between the DNN model (Table 5.1) and the SVM model (Table D.1), the same patterns emerge. The 3Di composition gave the highest and most

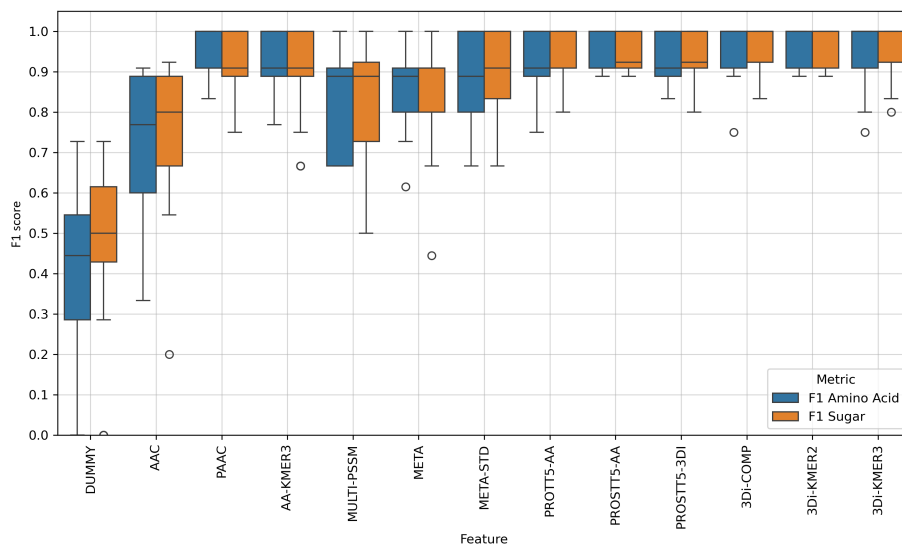


Figure 5.2: Distribution of 25 test set evaluation scores from five shuffled repetitions of a 5-fold cross-validation when distinguishing sugar transporters from amino acid transporters in *A. thaliana* with a deep neural network (DNN) trained on features that encode the amino acid sequence or protein structure. A short overview of the types of features used is listed in Table D.3. The box represents the area between the 25th and 75th percentiles of the score distribution, and has a line in the middle that shows the median. The whiskers extend to 1.5 times the length of the box at most. Any sample below or above the whiskers is classified as an outlier, and shown as a dot. As in Figure 5.1, the DL-based features in the form of embeddings and 3Di k-mers performed better than the previously developed META-STD feature. Interestingly, the amino acid 2-mer and 3-mers also led to good models when combined with a DNN. Overall, the DNN model produced more consistent results with higher median scores, as well as higher minimum scores for its two best-performing features. Again, the 3Di 2-mers matched the performance of DNNs trained with language embeddings.

consistent scores, closely followed by the ProstT5 embeddings, and closely behind these were the embeddings from ProtT5. Combining amino acid k-mer frequencies with 3Di k-mer compositions (COMB features) did not improve the results any further.

5.4.2 Categorization of Human Ion Channels

One type of transport protein that previous models struggled to classify was (monoatomic) ion channels. In contrast to the binding pockets that are commonly found in carrier proteins and many primary active transport complexes, these channels employ selectivity filters, i.e., narrow paths with multiple binding sites for ions and often for water molecules. Typically, these ion interaction sites are mainly made up of backbone carbonyl groups, although they are sometimes interspersed by charged residues, e.g., to orient substrate molecules in the channel. The selectivity of these channels is mainly determined by their width, and possibly by their flexibility [33]. This stands in contrast to the binding pockets of carrier proteins, where the residues need to match the chemical properties of atoms in the transported molecule in order to form non-covalent interactions [28]. On the other hand, several families of ion channels possess selectivity filters with conserved sequence domains. Since both ion channels and carrier proteins

Feature	Balanced Accuracy	F1 Amino Acid	F1 Sugar
DUMMY	0.475±0.120	0.400±0.216	0.477±0.207
AAC	0.754±0.155	0.742±0.156	0.754±0.171
PAAC	0.925±0.061	0.926±0.059	0.915±0.073
AA-KMER3	0.915±0.080	0.919±0.075	0.899±0.103
MULTI-PSSM	0.815±0.122	0.807±0.121	0.813±0.141
META	0.859±0.099	0.855±0.093	0.851±0.123
META-STD	0.889±0.091	0.883±0.097	0.890±0.096
COMB-KMER1	0.917±0.084	0.909±0.100	0.915±0.085
COMB-KMER2	0.945±0.057	0.945±0.058	0.942±0.064
COMB-KMER3	0.945±0.052	0.947±0.051	0.940±0.058
PROTT5-AA	0.918±0.063	0.910±0.071	0.923±0.058
PROSTT5-AA	0.944± <u>0.047</u>	0.941±0.050	0.946± <u>0.046</u>
PROSTT5-3DI	0.937±0.051	0.933±0.053	0.938±0.053
3Di-COMP	<u>0.959</u> ±0.056	0.953±0.065	<u>0.963</u> ±0.049
3Di-KMER2	<u>0.959</u> ± <u>0.047</u>	<u>0.958</u> ± <u>0.049</u>	0.959±0.048
3Di-KMER3	<u>0.959</u> ±0.067	0.955±0.074	<u>0.963</u> ±0.061

Table 5.1: Mean and standard deviation of 25 evaluation scores from five shuffled repetitions of a 5-fold cross-validation when distinguishing sugar transporters from amino acid transporters in *A. thaliana* with a deep neural network (DNN) and different features derived from the amino acid sequence or protein structure. The best results are underlined.

can contain conserved sequence motifs and conserved structural domains that are directly associated with their substrate specificity [33, 28], we explored how well the sequence- and structure-based feature encodings described in this study can capture these differences in ion channels, and how they compare in terms of classification performance.

5.4.2.1 Dataset Analysis

To estimate how well the new features and models are able to encode the substrate specificity of monoatomic ion channels, we created datasets of all human channel proteins transporting either calcium, chloride, or potassium. GO annotations inferred by electronic annotation were not included in the dataset. Proteins were filtered for those in the manually reviewed Swiss-Prot subset of UniprotKB, which also had evidence for the existence of the protein sequence, and known gene names. Sequence fragments were excluded. After dataset creation, where we only kept experimentally verified data, we were left with 105 calcium channels, 100 potassium channels, and 72 chloride channels, respectively.

According to GO terms annotated with experimental evidence, the channels possess a wide range of gating mechanisms. Out of 105 calcium channels, 36 are known to be voltage-gated, 26 are ligand-gated, 10 are store-operated [443], and 1 is stretch-activated. Only 11 out of 72 chloride channels are voltage-gated, whereas the 40 ligand-gated channels are activated by a wide range of molecules, including calcium (19 channels), GABA (14 channels), glycine (5 channels), and ATP and glutamate, which both activate 1 channel each. Four volume-sensitive and two pH-sensitive Ca^{2+} channels were in the dataset as well. The vast majority of potassium channels are voltage-gated, 78 out of 100 in total. Ligand-gated potassium channels are activated by calcium (11 channels), sodium (2 channels), as well as chloride and ATP, which activate 1 channel each. Further, three channels are mechano-sensitive. This means that potassium channels were the most extensively studied and annotated channel type in our dataset, by a large margin.

The overlaps between the classes were small, with five proteins that carry both Ca^{2+} and K^{+} , and one channel that transports Ca^{2+} and Cl^{-} simultaneously. We decided to exclude these proteins from the respective binary classification tasks, as their numbers would be too small to provide enough information to a multi-output model, or to add them as a third class. One channel in each of the three substrate classes did not have a 3D structure available in AlphaFoldDB, and was therefore excluded from the dataset as well.

Finally, we applied a sequence identity threshold of 70% to each substrate class. This left us with three binary classification tasks: 100 calcium channels and 58 chloride channels, 100 calcium channels and 84 potassium channels, and 88 potassium channels against 59 chloride channels.

In summary, all three substrate classes contained channels from a wide range of families and mechanisms, and from different evolutionary origins. This means that the evaluation of ML models on these datasets reflects their ability to distinguish between the actual substrates that are selected by the central cavity.

5.4.2.2 Results for SVM and DNN Models

The SVM and DNN models were evaluated on the same datasets, with the same 25 cross-validation splits. This was achieved by repeating a 5-fold cross-validation five times, shuffling the dataset after each CV, and starting with the same random seed. The feature datasets were exported from the SVM evaluation after all filtering was finished, and subsequently imported directly into the DNN evaluation methods.

First, we compared the previously best-performing model, namely, an SVM model with ANOVA F-score feature selection and META-STD protein fingerprint feature, to a DNN trained on the same feature, as well as an SVM and a DNN trained on amino acid embeddings (see Figure 5.3a). While switching the model to a deep neural network increased the median score for $\text{Ca}^{2+}/\text{Cl}^{-}$, as well as the maximum and minimum score for the other two tests, a larger effect came from the new feature, which improved minimum scores in all cases and median scores in most. After replacing the META-STD feature with ProtT5 embeddings, the mean and standard deviation of the SVM model were improved

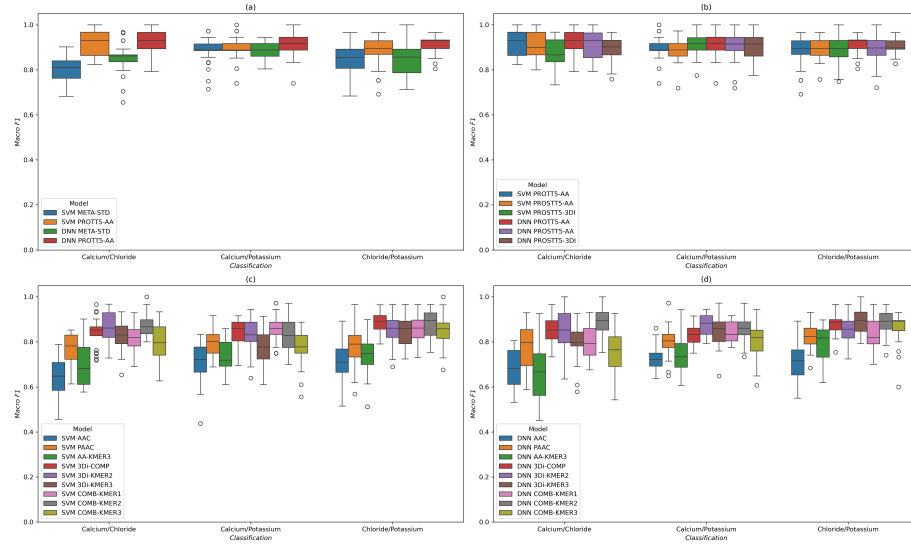


Figure 5.3: Distribution of macro-averaged F1 scores across five shuffled repetitions of a 5-fold cross-validation. Either a support vector machine (SVM) with feature selection or a deep neural network (DNN) were trained to distinguish between human ion channels in binary classification tasks. All tests were carried out on the same splits of the same datasets. The scores were calculated on the test set of each CV split. The boxes show the range between the 25th and 75th percentiles, and the whiskers extend in length to 1.5 times the range between these percentiles at most. Any score that is further away is visualized as a dot, denoting an outlier. The features are categorized according to feature generation methods and underlying data in Table D.3. In (a), we compared our previously developed META-STD feature to amino acid sequence embeddings calculated with ProtT5, and also tested the impact of replacing the SVM model with a DNN. The differences between two different amino acid embeddings and a 3D structure embedding are shown in (b). While the DNN model improved the median and minimum scores for several features, the choice of feature generation algorithm had a larger impact on model performance. In (c,d) we compare all features derived from k-mer frequencies in the protein sequence and/or the 3Di structure sequence with an SVM and DNN model, respectively. Each classification task seems to have a different combination of features and model that leads to the best performance, suggesting that an ensemble method would likely be necessary in order to compete with the performance of protein language embeddings.

by 0.07, while the standard deviation was lowered by 0.01 when distinguishing between calcium channels and chloride channels (see Table 5.2).

Next, we explored how the additional amino acid and 3D-structure embeddings we encoded with the ProstT5 model performed in comparison to the ProtT5 embeddings, both with the SVM pipeline and with the deep learning model (see Figure 5.3b). Overall, the performance of these features on ion channels was quite similar. In the $\text{Ca}^{2+}/\text{Cl}^{-}$ classification, the 3Di features led to lower median scores, whereas DNN led to slight improvements compared to the SVM. For $\text{Ca}^{2+}/\text{K}^{+}$, the 3Di embedding feature managed to improve the scores of the SVM model, delivering very similar results to the DNN model with amino acid embeddings. The best minimum score and lowest standard deviation were observed for the 3Di/DNN model for distinguishing between chloride channels and potassium channels, whereas the best median score came from the DNN with ProtT5 embeddings. According to the mean and standard deviations in

Model Substrates Feature	DNN			SVM		
	Ca/Cl	Ca/K	Cl/K	Ca/Cl	Ca/K	Cl/K
DUMMY	0.522±.088	0.477±.062	0.498±.088	0.459±.084	0.505±.080	0.471±.078
AAC	0.684±.087	0.725±.053	0.712±.088	0.645±.084	0.703±.086	0.705±.081
PAAC	0.775±.094	0.805±.075	0.828±.063	0.767±.067	0.795±.059	0.787±.083
AA-KMER3	0.663±.122	0.752±.080	0.798±.078	0.700±.097	0.743±.076	0.741±.077
3Di-COMP	0.844±.060	0.831±.044	0.876±.050	0.839±.068	0.832±.055	0.885±.043
3Di-KMER2	0.854±.082	0.877±.049	0.845±.058	0.860±.068	0.832±.075	0.854±.065
3Di-KMER3	0.795±.078	0.855±.076	0.894±.060	0.825±.066	0.777±.076	0.845±.061
COMB-KMER1	0.798±.077	0.848±.049	0.837±.071	0.817±.063	0.857±.063	0.864±.061
COMB-KMER2	0.889±.063	0.861±.054	0.882±.059	0.880±.056	0.837±.066	0.880±.060
COMB-KMER3	0.755±.095	0.801±.080	0.861±.076	0.798±.082	0.773±.079	0.847±.058
PSSM-50-1	0.769±.084	0.834±.067	0.791±.071	0.755±.061	0.808±.063	0.783±.071
PSSM-50-3	0.796±.052	0.863±.061	0.811±.075	0.803±.072	0.839±.056	0.806±.069
PSSM-90-1	0.769±.105	0.776±.050	0.778±.083	0.739±.070	0.742±.045	0.759±.067
PSSM-90-3	0.784±.084	0.773±.062	0.781±.087	0.739±.070	0.742±.045	0.759±.067
MULTI-PSSM	0.780±.077	0.833±.072	0.785±.082	0.783±.057	0.797±.052	0.770±.077
META	0.782±.072	0.803±.084	0.812±.065	0.826±.062	0.812±.061	0.784±.082
META-STD	0.851±.069	0.891±.038	0.850±.080	0.807±.055	0.884±.061	0.839±.072
PROTT5-AA	0.921±.059	0.915±.056	0.918±.043	0.922±.058	0.899±.054	0.889±.056
PROSTT5-AA	0.909±.065	0.903±.065	0.898±.061	0.911±.062	0.893±.054	0.898±.045
PROSTT5-3DI	0.894±.057	0.908±.053	0.908±.038	0.878±.068	0.911±.050	0.890±.061

Table 5.2: Means and standard deviations of macro-averaged $F1$ scores across five shuffled repetitions of a 5-fold cross-validation, which led to 25 test set scores in total for each test. Both the deep neural network (DNN) and the support vector machine (SVM) model were tested across three classification tasks, where the goal was to distinguish between two classes of human ion channels. All three substrates are shortened to their chemical symbols (Ca = calcium, Cl = chloride, K = potassium). The dummy feature consisted of 1024 random numbers drawn from a uniform random distribution for each protein, and showed the performance of a feature with no useful information. The best mean and standard deviation for each test are underlined.

Table 5.2, the embedding models led to the best performance in most cases, with the META-STD and 3Di frequency features often close behind.

Furthermore, we explored how the k-mer frequency features, which can be directly calculated from amino acid or 3Di sequences, compared to each other. For this comparison, we trained an SVM model, as shown in Figure 5.3c, and a DNN model, shown in Figure 5.3d, again on the same 25 train/test splits that were used for all four plots in Figure 5.3. The noteworthy point is that the amino acid k-mer frequencies delivered the lowest performance across all models and classification tasks, with only the PAAC feature, which contains 2-mer frequencies, delivering adequate results. This was expected, as it was the reason why we combined the AAC and PAAC with Multi-PSSM into the

META-STD feature, which led to better-performing models than its separate parts (as can also be observed in Table 5.2). Table 5.2 also shows that PSSMs calculated from more distantly related proteins, e.g., those found by three iterations of PSIBLAST on Uniref50, consistently led to better-performing features than PSSMs derived from closer homologues, such as those found with one iteration on Uniref90, suggesting a possible connection to model performance. The 3Di sequence features gave better predictive models in all three cases when paired with either of the two ML algorithms, similarly to what we observed when categorizing *A. thaliana* carrier proteins in Section 5.4.1.2. Combining the information from both sequence types led to better results in most tests, especially when combining PAAC with 3Di 2-mers (COMB-KMER2), which was among the best features across all tests and for both the SVM and DNN model.

Again, we found that the difference lies mainly in the choice of features and less so in the model itself, at least when comparing the SVM and DNN models from this study. The other consistent observation was that each combination of substrate classes prefers a particular type of feature over another. Therefore, a purely sequence-based method that is competitive with protein embeddings will likely have to be an ensemble approach that creates different models from both amino acid and structural sequences, and then selects the best one based on training data performance.

5.5 Discussion

In this study, we compared our previously developed methods for membrane transporter substrate prediction [349][376], where we trained support vector machines (SVMs) on ML-features derived from amino acid sequence and evolutionary information, against new feature extraction and classification techniques empowered by advancements in deep learning and natural language processing.

We used the encoder from the pre-trained ProtT5 transformer model to turn amino acid sequences into fixed-size word embeddings, which were then averaged across the protein sequence to create a numerical representation. Furthermore, we retrieved 3D structural models for the considered protein sequences that were predicted by the AlphaFold transformer model, and then used Foldseek to turn the three-dimensional atomic configurations into 3Di sequences, which represent protein structures with an alphabet of 20 characters. These 3D-structure sequences were then turned into embeddings with the bi-lingual ProstT5 model that has been trained on both types of sequences, and which we also used to generate a second dataset of amino acid embeddings. We also directly calculated ML-features from the 3Di sequences by turning them into vectors of k-mer frequencies, and also combined those with k-mer frequencies of amino acids. Finally, we trained our own DNN with TensorFlow, and compared the performance to our previously developed SVM model. To estimate the amount of information that is provided by the features, we also included a feature that was drawn from a uniform random distribution.

We tested our models by predicting the substrates of carrier proteins in *A. thaliana*, and of ion channels in human, in a cross-validation approach. When analyzing the dataset of sugar and amino acid transporters in *A. thaliana*, we

found that each substrate class contained proteins from multiple protein (super-)families with no or little evolutionary relation to each other. This indicates that our models were in fact predicting the substrate, and not, for example, the superfamily fold.

When training and evaluating an SVM on the data, the feature made up of 3Di k-mer frequencies surprisingly outperformed the protein embeddings, both in the mean score and the standard deviation across the 25 test sets. The 3Di 2-mer frequencies delivered particularly good results, with a mean balanced accuracy of 0.971 ± 0.044 , which was 0.061 higher than the ProtT5 amino acid embeddings (see Table D.1), as well as the highest minimum score (Figure 5.1). Combining frequencies of amino acid k-mers with the 3Di features produced similar results as the three embeddings, of around 0.90.

Our automatic outlier detection method classified three amino acid transporters as divergent from the rest of the data, which made sense upon further inspection. Removing these three proteins allowed us to test the performance on a “clean” dataset. This improved the average evaluation scores of the ProstT5 models compared to the ProtT5 embeddings (Table D.2), bringing the performance of the 3Di embeddings close to that of the 3Di 2-mer frequencies (Figure D.1).

Finally, we tested the TensorFlow DNN model on the automatically generated dataset (i.e., without removing any outliers), and found improvements to most features in terms of their average performance, as well as their consistency across the different training and test sets. Again, the 3Di 2-mer model led to the best performance, with a balanced accuracy of 0.959 ± 0.047 , closely followed by the ProstT5 embeddings, and the combined 3Di and amino acid frequencies, with better mean and standard deviation compared to the ProtT5 embeddings (Table 5.1). Both the ProstT5-AA and 3Di 2-mer models delivered very stable models, with the lowest observed test scores around 0.89 (Figure 5.2).

For the task of distinguishing three types of human ion channels in three binary classifications, the same three features delivered the best results, but their evaluation scores were arranged in the opposite order. This time the embeddings led to mean macro-F1 scores between 0.90 and 0.92, followed by combined PAAC and 3Di 2-mers features of around 0.84–0.89, and finally the 3Di 2-mer feature on its own, falling between 0.83 and 0.88 (Table 5.2). Replacing the SVM with a DNN led to relatively small differences in the mean and standard deviation of the scores across the 25 tests, while the bigger difference came from the choice of the features themselves (Figure 5.3). As for the *A. thaliana* dataset, we analyzed the protein families and co-occurring GO terms in the dataset, and found multiple different gating mechanisms and protein families in each of the three substrate classes.

An interesting aspect when considering the performance of the 3Di frequencies is the time it takes to calculate each feature. Deriving a single PSSM profile from Uniref50 or Uniref90 with PSI-BLAST can take several minutes, even on a fast compute server, and the Multi-PSSM feature (and therefore also the META-feature) requires four PSSMs for each protein. Calculating protein embeddings locally requires a similar amount of time when using the CPU, but the process can be accelerated to less than a minute per protein with a CUDA-compatible graphics card and a sufficient amount of available video memory. On the other

hand, the 3Di sequences for an entire proteome can be calculated with Foldseek in less than one second, and the only requirement is a database of protein structure files, which can be created or downloaded through the same tool. The 3Di k-mer feature could therefore be a faster alternative to embeddings or to features that encode evolutionary conservation, e.g., for large-scale analyses.

In conclusion, we found that deep learning delivers the necessary tools to both encode and accurately predict the substrate specificity of membrane transporters, improving on existing approaches such as SVMs trained on amino acid frequencies and evolutionary conservation profiles. However, the exact ML model and feature required depend on the substrates in question. This is due to the fact that transporters of one substrate can include many different protein families, domains, and 3D folds, with different or common evolutionary origins. Finding a model that captures the exact information needed to determine the substrate specificity while also tuning out confounding variables, such as a common evolutionary origin between some proteins, likely requires an ensemble approach that selects a model and a type of feature based on training data. Furthermore, combining data from several organisms, either directly or through transfer learning, could allow for the method to be applied to datasets with an insufficient number of training samples.

Chapter 6

Conclusion and Outlook

6.1 Summary

6.1.1 Introduction and Background

The goal of this thesis was to develop predictive methods that can assign membrane transporter proteins to one or more substrate classes, such as hexose, amino acid or potassium.

As we found in Section 2.1 and summarized in Section 2.1.3.7, proteins from one substrate class can belong to multiple protein families or superfamilies, with different structural folds and/or protein sequence domains. A single (super-)family can also encompass transport proteins of different substrates, and some transporters carry multiple types of molecules simultaneously or in an alternating fashion. Therefore, methods based on homology of sequence and structure can deliver inconsistent results for this task. They would be suitable if the transporters in each individual substrate class share a common fold or sequence motif that is not found in any other class, but that is typically not the case. More commonly, a super-family that shares a particular fold and transport mechanism develops first, and then adapts to a different substrate molecule at a later stage of evolution, through a small number of amino acid substitutions, or by evolving to bind different sub-domains. On the other hand, many transporters contain binding pockets that temporarily bind the substrate with a binding energy that must fall within an optimal range. This means that the size of the pocket, as well as the chemical composition of substrate-facing amino acid residues, are often highly similar between two transport proteins of the same molecule. If we therefore create a numerical vector for each protein that describes its functional characteristics, three-dimensional shape and fold, sequence conservation, and the chemical composition of amino acid residues, then we can employ methods from mathematical statistics and machine learning to determine those vector dimensions that let us encode these differences, in order to assign a protein into one substrate class or another, independently of evolutionary relationships.

When exploring the diverse landscape of transmembrane transport proteins, we also came across a small number of proteins that will likely be incompatible

with these methods. There are ABC transporters where the substrate selection is carried out by a substrate binding protein (SBP) that floats freely in the periplasm, and the transporter itself only contains a mostly non-selective “greasy slide” that opens up after docking of the SBP (Section 2.1.3.1). Some types of multi-drug transporters contain cavities with multiple overlapping binding pockets that allow for the selective transport of many different substrates, potentially making it harder to find common patterns (e.g., Figure 2.7 and Section 2.1.3.2). Some outer-membrane pores (OMPs) only select their substrate based on molecule size (Figure 2.14a), while others show unique mechanisms for substrate selection, such as the ball-and-chain mechanism of TonB (Figure 2.14c). This, along with their uncommon β -barrel structure that is vastly different in shape and amino acid composition from the typical α -helical bundle shared by most transmembrane proteins, makes them unsuitable candidates for our transporter classification models. While we do incorporate transmembrane transport domains of primary active transport complexes if they contain binding pockets, as is the case for some ABC nutrient importers of *E. coli* (see Figure 2.6) and some ATPases (e.g. the P-type ATPase pump in Figure 2.10), the main focus will be on carriers and ion channels.

When constructing the dataset, we wanted a diverse selection of transporters of each substrate that includes different folds and sequence motifs, as that allows us to find the similarities within each substrate class, along with the differences to transporters of other types of molecules. There is, however, an upper limit to the possible variety within each class. For example, it probably does not make sense to put proton-driven amino acid antiporters similar in shape and function to the sodium-driven LeuT shown in Figure 2.13, into the same “proton transporter” substrate class the rotary ATPase proton pump shown in Figure 2.9. Their differences will likely be too great, potentially introducing noise and making it harder to find the exact similarities and differences that are related directly to the function. This would compound the effects of the low sample counts that we often get with biological datasets. A more sensible approach would be, for example, to include carriers from different superfamilies in each class, such as MFS and LeuT (see Figures 2.13 and 2.11,2.12), along with suitable transmembrane transport domains from primary active transporters (e.g., MalF from Figure 2.6), and then train a second model for the classification as ion channels (such as the potassium channel shown in Figure 2.16), possibly using features that more emphasis on structural differences.

In Section 2.2, we first listed the available data sources for various types of biological data, including protein sequences and 3D structures. We also explored how protein annotation data is arranged into ontologies, and how we can determine if annotations have been experimentally verified by a published study, or only computationally predicted. We described the methodologies behind available command line tools, e.g. for sequence clustering or sequence database search. In Section 2.2.2, we then introduced a range of statistical methods that allow us to find differences and similarities between groups of proteins, and estimate the distributions of their attributes. This included independent t-tests, ANOVA, multiple-hypothesis correction, correlation analysis, tests for normality, as well as non-parametric, rank-based alternatives to these methods if the data does not follow an approximate normal distribution.

Then, we explained in detail how support vector machines (SVMs) are trained, including the underlying mathematical optimization problems and theorems, in Section 2.2.3. Furthermore, we introduced extensions to the original SVM approach that overcome its limitations as a binary and linear classification algorithm. On one hand, this included methods for applying SVMs to multi-class and multi-label tasks. On the other hand, we also described the mathematical background behind coordinate transformations and kernel functions that allow for training on datasets that are not linearly separable, with special focus on the radial basis function (RBF) kernel.

Finally, we described methods for reducing the number of feature vector dimensions to those that are useful to the classification task at hand, for avoiding overfitting on the training- or test data during the optimization process. This included principal component analysis (PCA), univariate feature selection with ANOVA F-scores, as well as L1 regularization. For evaluation and hyperparameter optimization, we discussed various performance metrics, with their respective advantages and disadvantages, and ideal applications. Finally, we explained how these metrics can be calculated in a cross-validation approach, to avoid overfitting and improve stability and generalization.

6.1.2 Outline

The project was comprised of three main parts. First, we needed to automatically assemble training datasets from published literature, with parameters that allow for a trade-off between quantity and quality of samples. These methods aim to create datasets with a sufficiently high variety of proteins from different families, ideally with no confounding variables, and assign the transporters into known substrate classes for training. Then, we needed to find multi-dimensional numerical protein encodings that are related to the functional differences between the substrate classes. These encodings were derived from sequence, 3D structure or evolutionary information. Finally, we developed machine learning pipelines that can filter the provided information for those vector dimensions that are useful for the classification task at hand, and can then accurately separate the groups of proteins from each other, while delivering stable and consistent results on previously unseen data. The code was released as a Python package that allows for easy access to these methods through an API, with example usage in Jupyter notebooks.

6.1.3 Chapter 3: Developing datasets, features and models

We started with a model for telling apart sugar carriers from amino acid carriers in *A. thaliana*, as a proof of concept. In this initial version of the dataset pipeline, we derived protein annotations from the Uniprot Keywords [444] database. As most keywords are relatively broad in their description, we created more specific annotations by calculating intersection sets, e.g. of proteins annotated with the location keyword “Transmembrane”, and the functional keyword “Sugar transport” (see Appendix B.2). While this did find many transporters, it also included some false positives, requiring careful filtering. This was due to the fact that functional keywords are related to the broader “biological

process” annotations from the gene ontology (GO) [47], rather than the more precise “molecular function” aspect of GO. This meant that, e.g., single-pass membrane proteins that regulate transport by interacting with transmembrane transporters were also in the initial dataset, and had to be filtered out with additional logic that handles these edge cases. In the following chapters, we therefore changed the source for our functional annotations to GO, as that provides us with more control and precision when constructing datasets. We also considered using the transporter classification database (TCDB) [233] for training, but the number of available transporters was lower compared to the Uniprot-Keywords dataset, especially for *A. thaliana* (see Section 3.4.2.4).

Next, we studied published literature on numerical protein encodings for machine learning, in order to test and compare their respective performance when training models for substrate prediction. This included features derived from frequencies of amino acid types and dipeptides in the sequence (AAC and PAAC, respectively), averaged position-specific scoring matrices (PSSMs), amino acid properties retrieved from AAIndex [445], a feature based on genomic neighborhood and co-expression networks [375], and we also tried more traditional feature generation algorithms, such as one-hot encoding.

Along with these feature encodings, we evaluated the performance of a wide range of parametric and non-parametric machine learning algorithms, including support vector machines (SVMs) with different kernels, k-nearest-neighbors (KNN), random forests, perceptrons, naïve Bayes, SVMs with stochastic gradient descent (SGD-SVM), as well as various ensemble methods. Each algorithm was tested extensively, with pre-processing, full hyperparameter optimization and nested cross validation (see Section 3.4.6).

The first attempts at developing a ML model for membrane transporter substrate prediction centered around the co-expression-based feature introduced by Tran et al. [375]. The idea was to encode the functional GO annotations of co-located and co-expressed genes of transport proteins, and then associate that GO profile with the substrate class. We extended this method by creating and combining co-expression networks for multiple human tissues, as well as for yeast and *E. coli*, from baseline gene expression data retrieved from GTEx [446] and GEO [447]. However, we found that the performance was too inconsistent, as the method expects the neighbor genes of the transporter gene on the genome or chromosome to be involved in some way in its transport function. While this is often the case for bacterial operons, there are many exceptions in more complex organisms, such as mammals or plants. Part of the reason could be that operons coding for transport complexes often go through gene fusion events, where the individual proteins merge into two half-transporters, or into a single gene. Another reason for the inconsistent performance was that many transporter genes in these organisms simply did not have any neighboring genes that were also co-expressed, making them unsuitable candidates for the method.

The idea that led to the first models with good classification performance was to calculate position-specific scoring matrices (PSSMs) on larger sequence databases. Previous studies [40] only derived these evolutionary profiles from the much smaller, manually curated SwissProt subset of Uniprot, which were made up of around 250MB of sequence data at the time. We calculated PSSMs locally on the Uniref50 and Uniref90 datasets, containing 21.6GB and 70.1GB of

protein sequences, respectively. Furthermore, we tested the impact that different PSIBLAST parameters (i.e. the tool we used to calculate PSSMs, see Section 2.2.1.8) on the classification performance. These PSSMs were then averaged into fixed-size amino acid profiles that encode the conservation of individual amino acid types.

We achieved the best classification performance with a machine learning pipeline that first standardized the data, reduced the high number of feature dimensions with either principal component analysis (PCA) or feature selection based on ANOVA F-scores, and finally trained a SVM with an RBF kernel.

As we had calculated multiple PSSMs for each transport protein with different parameters, we also trained ML models on multiple PSSMs simultaneously. This “Multi-PSSM” feature encoded information derived from both closely- and distantly related proteins, depending on the respective parameters for the PSIBLAST software [299]. Finally, we combined the Multi-PSSM feature with general sequence information in the form of the AAC and PAAC, into the “META-feature”, which led to the best results for distinguishing sugar- from amino acid transporters in *A. thaliana*, when used as input for the ML pipeline.

With the META-feature, we had now found a series of data points related to protein function that contained the necessary information for telling apart the proteins from two substrate classes, independently of protein families. When calculating the feature importance scores in Figure 3.6, we found that amino acid types commonly observed in transmembrane helices were of little to no importance to the model, while charged residues commonly found in functional sites brought the most valuable information. The ML pipeline comprised of standardization, feature selection and RBF-SVM was capable of filtering this information for exactly those dimensions that produce accurate and generalizable models for the classification task at hand. Hyperparameter tuning with cross validation, along with testing on independent test sets, ensured that the model did not overfit on the training- or evaluation data. Clustering of the protein dataset at a sequence identity threshold of 70% aimed to counteract possible information sharing between training- and test datasets, by removing isoforms or close homologs of the proteins.

After we finished optimizing the proof-of-concept model on *A. thaliana*, we applied it to different organisms, and explored the possibility of adding substrate classes.

We started by creating evaluation datasets for three additional organisms: *E. coli*, *S. cerevisiae*, and human. During dataset analysis, we found that the substrate classes for the respective organisms also contained unique protein families that were not present in the species. For example, the outer membrane of *E. coli* included outer membrane pores (OMPs) with a β -barrel structure, which were not compatible with the rest of the dataset due to the vastly different amino acid composition compared to the transmembrane helices commonly found in transport proteins. After removing these five OMP proteins, the classification performance on the *E. coli* dataset improved considerably, with F1 scores above 0.9 during cross validation and on an independent test set (see Table 3.2).

The test scores for the yeast dataset were much lower compared to the cross validation during training. When looking at the sample counts in Table

3.1, we noticed that evaluation scores were correlating with the total number of available samples in each dataset. A low number of samples in the independent test set can lead to problems with stepping, where each additional true positive can change the accuracy score by a large margin. For example, the accuracy on a test set with only five samples would increase in steps of 20%. A small sample count in the training set can also cause the model to fit to patterns that are irrelevant to the classification task, which is referred to as overfitting.

To address both of these concerns, we first tested the model in a nested leave-one-out cross validation (LOOCV), and found that both the training and test scores improved considerably, possibly due to the larger number of samples in the training set, and the more stable calculation of average test scores (see Table 3.3). As a test, we randomly down-sampled the best-performing *A. thaliana* dataset, and found that the class-wise performance matched that of the yeast dataset after it was reduced to the same number of samples (Figures 3.4a,b). From the observations we made during these tests, we concluded that training datasets for membrane transporter substrate prediction need to contain at least 20 samples per class, after sequence identity clustering.

We tried to solve the problem of low sample count in several different ways. At first, we tested the possibility of training models on organisms for which we had achieved good results, and to then apply those models to datasets with fewer proteins. As can be seen in Tables 3.4 and 3.5, this did not work for any pair of organisms. When taking a closer look at the datasets, these results make sense, as there are several families of transporters that are mostly specific to one organism in particular, thereby influencing the decision function when trained on that organism. For example, the large SWEET family of 13 sugar exporters in *A. thaliana* only had a very small number of homologues in the other three organisms (only one in human), the inner membrane of *E. coli* contain ABC importers for sugars and amino acids that are not present in eukaryotes, and human transporters include the SGLT family of sugar carriers, which belongs to the a super-family that is mostly made up of amino acid transporters (see APC super-family in Section 2.1.3.2).

The previous observations led us to the idea of combining transporters from all four model organisms into one meta-dataset, as that would contain the protein families from all four species. We tested all three ML features, both individually and combined as a meta-feature, with a pipeline consisting of PCA for dimensionality reduction, followed an RBF-SVM. Interestingly, the multi-PSSM feature performed better in this case than the meta-feature, which also includes sequence information in the form of the AAC and PAAC. Limiting the model to the three eukaryotic organisms improved the training and test scores by an additional 0.03 and 0.025, respectively (see Table 3.6). We then took a closer look at the performance of the best model, and calculated precision and recall for either of the two classes. On both the training data and the independent test sets, the model achieved an F1 score of more than 0.93 for either class, across proteins from human, yeast and *A. thaliana*. This means that training on datasets that contain transporters from multiple organisms can solve the problems caused by low sample count in single-organism datasets (see Table 3.7).

The next step was to include more substrate classes. At first, we developed and tested a hierarchical assembly of SVM classifiers that was organized similarly to the transporter classification system by Saier [448] (see Section 2.2.1.7). The first classifier would predict the mechanism, such as channel or carrier or pore, then the second classifier would predict the super-family and/or fold, e.g. MFS or APC, and finally the last layer of classifiers would predict the substrate. This was supposed to make sure that large structural differences between groups of proteins were not an issue, by only predicting substrates within the same super-family, where the member proteins often have a similar topology. However, while each individual model achieved good accuracy, the respective scores after each step had to be multiplied to evaluate the entire model, in its accuracy for substrate prediction. For example, if each step had an accuracy of 90%, and the final substrate was determined after three steps, the model would have an overall accuracy of $0.9^3 = 0.729$. Furthermore, recent advances in protein structure prediction [21], along with membrane orientation prediction through MemProtMD [277] or PDBTM/TMDet [278], allow us to determine the mechanism and fold more directly (see Figures 2.12a,b and 2.13a,b).

The second attempt at a multi-class model involved a one-vs-one voting scheme, where pairwise SVM models were trained for all combinations of two substrates. As an initial test, we started by extending the previously developed model for distinguishing sugar- from amino acid transporters in *A. thaliana* with additional substrates. We found that adding potassium- and electron transporters, as described in Sections 2.1.3.5 and 2.1.3.6, respectively, led to a very good classification performance across all four classes 3.10. The optimal choice of substrate classes was the subject of Chapter 4, and will be summarized below. Interestingly, while using PCA to reduce feature dimensions worked well in the binary case, it was not as effective for the multi-class case, where feature selection with ANOVA F-scores delivered better results.

When evaluating the previous results, we noticed that the choice of best-performing feature largely depended on the classification task at hand. To explore this further, we implemented an ensemble method that automatically combined different feature datasets, and provides a hyperparameter that allows the best combination to be determined from training data, with grid search. The pipeline was then able to select any combination of the four PSSMs that we calculated for each protein, as well as the AAC and PAAC features. When creating 1,000 randomly drawn training-test-splits and training an ML pipeline with the custom transformer on each, we found that the model showed a clear preference for PSSM features calculated from more distantly related proteins, i.e. those derived from three iterations of PSI-BLAST on either Uniref50 or Uniref90. Only 25.6% and 6.7% of models included AAC or PAAC, respectively (see Section 3.4.5 for more details).

Finally, we compared different ML algorithms again, to estimate their performance in the multi-class case, together with the custom transformer for automatic feature combination. While the non-parametric algorithms, i.e. KNN and RF, did lead to models with clearly lower performance, an SVM with stochastic gradient descent (SGD-SVM), as well as a single Perceptron, matched a RBF-SVM model in its evaluation scores. Combining the Perceptron with elastic net regularization led to a very stable model for the multi-class dataset, with macro-

averaged training and test scores of 0.978 ± 0.006 and 0.969 ± 0.024 , respectively (see Section 3.4.6).

6.1.4 Chapter 4: Improving data sources and finding optimal substrate classes for any dataset

In Chapter 3, we had extended our model for sugar and amino acid transporters to include additional substrates, and found that the performance of these multi-class models largely depended the substrate classes that were added, and on the dataset. The goal of the second published manuscript in Chapter 4 was then to find a set of substrate classes that cover as many transmembrane transporters as possible, are as specific as possible in their functional description, and also contain enough proteins for training, i.e. at least 20 samples per class.

First, we replaced Uniprot Keywords, which we had previously used for functional protein annotations, with gene ontology terms. Keywords were less precise than GO terms from the “molecular function” (GO-MF) aspect of GO, and frequently required additional filtering to remove proteins from the dataset that were involved in pathways related to transmembrane transport, but did not carry out the function themselves. GO-MF also contains more precise functional descriptions. For example, whereas keywords only annotate proteins as sugar transporters, GO-MF also has child-terms for carriers of hexoses or of D-glucose.

As described in more detail in Section 2.2.1.5, the GO-MF terms are arranged hierarchically, in a tree structure, where paths lead back to the root node. Here, the “is_a” relation describes a logical connection between a more abstract and a more specific term; we removed any other type of relation from the tree. Furthermore, we only kept the sub-tree of GO-MF terms that describe a type of “transmembrane transporter activity”. When using experimental evidence from published literature to annotate this tree with the proteins from our filtered protein dataset, we observed that the sample count of a GO term decreases with increasing distance from the root node, as functions get more specific. While many proteins were experimentally verified to transport a more abstract substrate description, such as carbohydrate or sugar, few were explicitly annotated with the transport of a very specific sugar molecule, e.g. α -D-glucopyranose. This means that GO terms and their associated protein sets, there is a trade-off between the number of samples available for training, and the precision of the functional annotation. Many proteins are only annotated with a more specific GO term, but not with the more abstract terms that are connected to this term by “is_a” relations. We propagated these annotations upwards in the tree, which improved the sample counts for abstract classes considerably. For example, if a carrier is only annotated with “hexose transport”, and there is a directed path between “hexose transport” and “sugar transport”, then the carrier is also annotated with the latter. The resulting dataset creation pipeline provides several additional parameters that allows for a trade-off between sample count and data quality, e.g. by filtering for different evidence codes both in Uniprot and in GO. The parameters, along with the effect they have on the available number of proteins and GO annotations, are listed in Tables C.1 and C.2, respectively.

As there are hundreds of GO terms associated with transmembrane transport functions, and testing their performance as substrate classes did not show a

clear pattern, we implemented an optimization algorithm that approximates an optimal subset of annotations. One challenge was that many proteins were annotated with terms from multiple substrate classes. For example, there are various types of symporters and antiporters (Section 2.1.3.2) that transport the same cation along with different solutes, and this substrate class overlaps with ion channels for that cation (Section 2.1.3.5), as well as large protein complexes that act as ATP-driven ion pumps (Section 2.1.3.1). Telling these annotations apart from others by training ML models on the annotated proteins would prove challenging, due to the heterogeneity in the set of training samples.

We started by finding and implementing additional similarity measures between GO terms. The idea was to find a number of constraints based on these metrics that would predict a good classification performance between two annotations, and their annotated protein sets. The final annotation subset should have minimal redundancy, cover as many proteins as possible, prefer specific functional descriptions over more abstract terms, ensure that enough samples are available in each class for training, and allow for the training of stable and accurate models.

The pairwise similarity measures between GO terms included five different semantic similarity algorithms from the GOSemSim package [383]. Four of these graph-based algorithms measured GO similarity through information content (IC), which describes how specific or abstract a term is, based on how many proteins are annotated with it in a given dataset. The similarity calculation between two terms then centers around the IC scores their common ancestors [449]. In contrast, the Wang method, which we ended up including in the manuscript, is calculated solely from path lengths of GO terms to their common ancestors and to the root node [384].

We derived further similarity measures directly from the GO terms and their annotated protein sets (see Section 4.3.3). This included pairwise sequence identity scores calculated with the Needleman-Wunsch algorithm, which we aggregated into a single number for each pair of GO terms by taking the median, mean, maximum or minimum identity between the two protein sets. For the entire set of GO terms, this gave us four sequence identity matrices. Furthermore, the *overlap matrix* contained the size of the intersection set between the two protein sets annotated with each pair of GO terms. The degree similarity score was calculated from the difference in the total number of edges that were attached to two GO terms in the GO graph, and the similarity of in-degree and out-degree denoted the difference in the number incoming and outgoing edges, respectively. Finally, the adjacency matrix simply contained binary values denoting whether there is an edge between two GO terms.

Next, we took advantage of the fact that cross-ontology connections exist between the GO and ChEBI ontologies, with multiple types of relations. ChEBI is an ontology that organizes chemical entities of biological interest, and is further explained in Section 2.2.1.5. We retrieved the ChEBI terms for our GO-MF sub-tree of transmembrane transport annotations, and filtered them according to the relations. Only ChEBI terms that were labeled as primary transported substrates were kept in the dataset (see Section 4.3.3.4). From these molecule annotations, we then calculated chemical similarity scores for each pair of GO terms, from their associated substrates. The idea was that chemically distinct

molecules also lead to different chemical compositions in transporter binding pockets and sequences, possibly making it easier for sequence- and structure-based ML models to tell the respective substrate classes apart. We calculated four molecular fingerprints from each molecule and compared them with the Tanimoto coefficient, as described in Section 4.3.3.4.

Finally, we trained 24 different ML pipelines for each pair of GO terms, to cover different methods for pre-processing, sequence clustering, dimensionality reduction and feature selection. We also compared different approaches for dealing with classification tasks where the classes are overlapping, testing multi-label and multi-output classifiers.

Now that matrices with pairwise similarity- and evaluation scores were assembled for all GO terms, we employed statistical testing and correlation analysis to find connections between them (see Sections 2.2.2 and 4.4.4.3). Both of these analyses led to the conclusion that out of the previously calculated similarity measures, the strongest predictors for classification performance were derived from pairwise sequence identity. The mean sequence identity was the only metric that was able to predict classification performance with a statistically significant p-value of $p = 0.0306$ after multiple-hypothesis correction, when performing a t-test between GO terms with high and low ML evaluation scores. The highest and lowest correlation coefficients between F1 scores and similarity measures were achieved by the minimum and maximum sequence identity, respectively, suggesting that a good pair of functional annotations for training binary ML classifiers should not be too similar, but also not too dissimilar from each other, in terms of their protein sequence identity (see Table 4.1). As several similarity scores were not normally distributed according to the Shapiro-Wilk statistic, and had therefore not been included when performing the independent t-tests, we then performed two one-tailed Mann-Whitney-U tests, in order to test a broader selection of similarity measures. After p-values were corrected with the Benjamini-Hochberg method, we additionally found the GO semantic similarity, the absolute difference in sample count and the maximum sequence identity to be statistically significant predictors of lower classification performance, while higher F1 scores were predicted by both the average and the minimum sequence identity.

Unfortunately, the chemical similarity scores were only available for a small number of GO terms. This was due to the fact that many abstract ChEBI terms, such as “sugar” or “hexose”, were not linked to a specific chemical structure, which is a requirement for calculating molecular fingerprints. Only GO terms that were linked to more specific ChEBI terms, such as “D-glucopyranose”, could be compared to each other with pairwise Tanimoto coefficients. We tried to solve this problem by retrieving descendants of abstract ChEBI terms that were linked to chemical structures, in order to create a matrix of chemical similarity scores between the descendants of two ChEBI terms instead, which would then be aggregated, e.g. by taking the median value. However, we found that ChEBI is too complex for that approach to be viable, as it can contain circular paths, connections between different sub-graphs, and a very large number of edge cases that would need to be manually removed. For example, the ChEBI term for sugar has hundreds of child-terms, and each of those sub-graphs contains several molecules that we would not want in our dataset, such as open-

chain sugars, multiple stereoisomers, variants containing carbon isotopes in different places, and more. Filtering these molecules for those that are relevant to each transport function, while also having high bio-availability in the organisms or tissues we are interested in, would be a very time-consuming task, and defeat the premise of creating an automated pipeline. This is especially the case for the large number of lesser-known substrate classes, whereas finding the correct molecules for classes such as sugar or amino acid would be relatively trivial. One such effort has recently been undertaken by Goldmann et al. [27], where 460 human proteins from the SLC super-family (see Section 2.2.1.7) were annotated with the ChEBI terms of their substrate molecules by a team of 16 researchers, who read through the published studies for each transporter. Based on the annotated molecules, the proteins were assigned into nine substrate classes, which were represented by abstract ChEBI terms.

After analyzing the results from the comparative analysis, we subsequently implemented and tested a number of optimization algorithms that aimed to find an optimal subset of GO terms based on constraints around similarity scores. The best method, which was included in the manuscript, created a training dataset, where it iteratively removed the GO term that had the highest negative impact on the average pairwise evaluation score, while also making sure that enough proteins are still covered by remaining annotations. The distance to the root node was used as a tie breaker. The algorithm is described in more detail in Section 4.3.4 and Appendix C.2. The full pipeline is shown as a flowchart in Figure 4.1. In a dataset of all membrane transporters in yeast, the algorithm reduced the number of GO-MF annotations from 287 to 11, while only losing 1.9% of proteins. The eleven classes had a median pairwise F1 score of 0.87. In a meta-organism dataset encompassing proteins from human, yeast, *E. coli* and *A. thaliana*, the number of GO terms related to transmembrane transport was reduced from 695 to 49 while keeping 96% of all proteins. The mean pairwise overlap between these 49 classes was only 2%, and their median pairwise F1 score was 0.92 ± 0.10 . Reducing the number of classes any further required us to remove more annotated proteins from the dataset. At a protein coverage of 67%, we found 30 classes with a mean overlap of only 1%, and the median pairwise F1 score increased to 0.95 ± 0.06 . This underscores our previous observation that one supervised ML model will likely not be capable of accurately classifying all known membrane transporters, including channels, carriers and ATP-driven pumps, simultaneously.

In conclusion, we introduced an optimization algorithm that allows for the reduction of redundancy and overlap within GO-MF annotation sets, without losing much information in the process. The smaller number of annotation, with low redundancy and overlap, simplifies the training of supervised learning models considerably. We also implemented a new dataset creation pipeline for automatically assembling protein sets based on molecular function annotations, which are, in turn, backed up by experimental data from published studies.

6.1.5 Chapter 5: Can recent advances in deep learning improve our features and models?

In the meantime, deep learning (DL) methods had been introduced to bioinformatics and successfully applied for various tasks, as we summarized in Section 5.2.3.

There were two applications of DL models that we were particularly interested in. First, the accurate protein 3D structure prediction with AlphaFold 2 [21] and similar tools opened up the possibility of training ML models on the three-dimensional folds of proteins, and not just on sequences, providing additional information to the model. And second, the recent advances in natural language processing that led to large language models (LLMs) had been applied to proteins, by interpreting each amino acid sequence as a sentence, each residue in that sentence as a word, and the 20 types of amino acids encoded by standard genetic code as the underlying dictionary.

Additionally, we wanted to train a deep neural network model and compare it to the previously developed SVM pipeline. In Chapter 3, we had already shown that a single-layer perceptron, which is the core building block of neural networks, was able to match the performance and consistency of a SVM. Furthermore, we tested the various types of layers and optimizers available in deep learning libraries, with the goal of replacing the feature selection in the ML pipeline with an internal regularization method.

Our first attempts at encoding protein structural models from AlphaFold 2 into fixed-size numerical vectors for supervised learning revolved around distance maps and contact maps. These quadratic matrices contain $n \times n$ dimensions for a sequence of length n . Distance maps list the distances between C_α atoms for each pairing of amino acids in the sequence, whereas contact maps convert these distances to binary or integer values, based on distance thresholds. We tested several methods for turning these quadratic matrices with variable, protein-dependent dimensions into machine learning features of constant length. This included padding of lower-dimensional maps, converting the matrices to bitmap images to then apply image convolution operations, and interpreting contact maps as adjacency matrices that represent graphs, to then use them as input for a graph convolutional network (GCN), similarly to the DeepFRI model for general protein function prediction [351].

We then discovered the FoldSeek tool [42], which turns protein structures into “3Di sequences” that are the same length of the underlying amino acid sequence, and use the same alphabet of 20 letters. These 20 states encode the backbone angles around each amino acid and around its nearest neighbor in the structure, along with the distance in both sequence and structure between the two (more detail in Section 5.2.4). These structural sequences allow methods developed for amino acid sequences, such as clustering, local and global alignments, as well as BLAST-like searches in 3Di sequence databases, to be applied to the 3D structures of proteins. We converted our protein structural models from AlphaFoldDB [21][275] into 3Di sequences with FoldSeek, and then turned them into fixed-size vectors by calculating normalized k -mer frequencies for $k \in \{1, 2, 3\}$.

As described in Section 5.3.8, we encoded both types of sequences with protein language models (pLMs). Amino acid (AA) sequences were first converted to per-protein embeddings with ProtT5 [378]. Conversely, the ProstT5 model [434] was trained to convert between amino acid and 3Di sequences, on a dataset that contains information from both. We created two additional embeddings for each protein with this model, from its AA sequence and its 3Di sequence, respectively. The 3Di sequences were calculated with FoldSeek, from structural models retrieved from AlphaFoldDB.

First, we compared the performance of these new features against the previously introduced encodings from Chapter 3. Again, we created a dataset of sugar- and amino acid transporters in *A. thaliana*, but this time with the GO-MF pipeline from Chapter 4. We used the strictest parameters when creating the dataset, to only retrieve proteins and annotations that are directly backed up by experimental evidence from peer-reviewed studies.

After thoroughly analyzing the resulting dataset in Section 5.4.1.1, we first tested both the new and the old features with our previously developed pipeline, which consisted of an RBF-SVM and feature selection through ANOVA F-scores. Table D.3 provides an overview of all the feature generation algorithms. As shown in Table D.1, the DL-based features brought substantial improvements over the encodings derived solely from protein sequence and evolutionary information that we introduced in Chapter 3. The new features also led to improved consistency across different training- and test sets, with minimum evaluation scores that were much closer to the median (see Figure 5.1). Interestingly, the 3Di k-mers gave us a classification performance that was very similar to the pLM embeddings, surpassing the previously best-performing PSSM- and META-features, while being much faster and less resource-intensive to calculate. Whereas creating the embeddings took several hours on a modern graphics card with 16GB of video memory, and calculating four PSSMs for each protein took several days on a CPU with 80 threads, the 3Di sequences for the four proteomes we considered for the study were calculated in less than a second, on a regular desktop processor.

Next, we tested replacing the SVM model with a deep neural network (DNN), and evaluated both on the same data. After trying various DNN configurations, with different types of layers, loss functions and optimization algorithms, we arrived at a feedforward neural network (FNN) with three densely connected layers, interspersed with dropout layers, where 50% of the connections are randomly severed during training. This introduces regularization to the model, and allows us to no longer rely on external methods for reducing feature dimensions, such as feature selection or PCA. We also tested the viability of dynamically creating neural networks based on the dimensions of the training dataset, determining the number of input nodes and/or layers from the number of features or samples. As this did not lead to a considerable change in classification performance, we selected the simpler static model instead, where all optimization- and training procedures now happen internally.

As shown in Table 5.1, the DNN led to further improvements in the average evaluation scores, with even lower variance (Figure 5.2). This was particularly apparent when comparing the performance of the pLM embeddings between the DNN and SVM models, where the minimum test scores were higher for the

DNN, with both 3Di k-mers and pLM embeddings achieving F1 scores of 0.89 or higher across five repetitions of a five-fold cross validation.

One type of transport protein where our previous models found limited success was that of ion channels. While the selectivity filters in these proteins often contain sequence motifs that a ML model could learn from, their selectivity can also rely on the exact distances between the amino acids that provide the binding sites for ions during transport, as we described in Section 2.1.3.5. We were therefore interested in the performance of the new model and features, when distinguishing between ion channels that belong to different substrate classes. We created a dataset of human ion channels that transport calcium, potassium and chloride. As the dataset analysis in Section 5.4.2.1 showed, these three substrate classes each encompass a variety of different protein families, and simultaneously had enough samples available for training.

When evaluating pairwise ML models for these three classes of ion channels, we found that the pLM embeddings delivered better results than most of the previously developed encodings, across the three models (see Table 5.2, Figure 5.3). As with the *A. thaliana* dataset, the DNN was able to deliver slightly higher results than the SVM and was, in most cases, more consistent across the 25 test cases. However, the largest difference came from the choice of feature generation algorithm. Some of the other features, particularly the META-feature from Chapter 3 and the 3Di 2-mer composition, did manage to match the embeddings in their evaluation scores, but the performance was more inconsistent between different datasets. In future studies, combinations of these features with embeddings of protein sequence and/or structure, possibly together with more complex DNN architectures, could improve the existing results further.

In conclusion, we can say that the recent advances in deep learning have allowed us to create better features and models for membrane transporter substrate prediction. We found that the training dataset and feature generation were more important for classification performance than the choice of ML model. Since the optimal selection of model and features depends on the classification task at hand, the optimal solution will likely involve an ensemble approach, similar to the automatic feature combination transformer that we tested in Section 3.4.5.

6.2 Conclusion and Outlook

At first, the task might seem simple. Amino acid sequences are strings of varying length, build from an alphabet of 20 different characters. They are assigned to one or multiple classes, depending on which substrate(s) they carry. Now, we need to find a mathematical model that associates these strings with the classes, and then we can predict the substrates of additional membrane transporters. However, when looking at the biological complexity behind transmembrane transport that we touched on with some examples in Section 2.1, and at the dataset analysis we performed in Chapter 4, it becomes clear that the solution is not as trivial, and requires careful selection of training samples, feature encodings, machine learning algorithms, and optimization methods.

On one hand, we needed a dataset of membrane transporters that were similar enough to be comparable, e.g. by including only ion channels, or only trans-

membrane proteins that have binding pockets, as is the case for most carriers and many multi-pass membrane domains of primary active transport complexes. On the other hand, we wanted protein sets that were not *too* closely related, as that would mean that we are actually predicting protein families or homologues/isoforms instead of substrate classes, which could also introduce information sharing between the training- and test sets, leading to biased evaluation results. Therefore, we employed a sequence identity threshold, and analyzed the training datasets for the number of motifs/domains/families that were present in each class before evaluation. Another task was to retrieve reliable annotation data for the proteins in UniprotKB, in order to find the membrane transporters, and then assign them to protein sets according to their transport mechanisms and transported substrates. We started with Keywords from the manually curated Swiss-Prot subset of Uniprot, but found that functional annotations were too broad. We then implemented a pipeline in Chapter 4 that took the entire Gene Ontology as its input, and filtered it down to an ontology of membrane transport functions. This pipeline employed parameters that allowed for a trade-off between quality and quantity of training data, by either solely relying on experimentally verified data from peer-reviewed studies, or also including predicted annotations or protein sequences.

We found that substrate classes can best be distinguished in two steps: First, an appropriate encoding is needed that contains the necessary information for telling apart one protein set from another. And second, since these relevant data points only make up a very small percentage of feature dimensions, we need sophisticated filtering methods for removing the unnecessary information and noise, without over-fitting on the training data in the process. To this end, we compared, combined, and expanded on algorithms for protein feature encoding, and introduced a new method that encodes protein structural models with deep learning, in Chapters 3 and 5. Our first ML pipeline used a training dataset to find an optimal number of features or principal components to keep, with feature selection or PCA, respectively, and then trained a support vector machine (SVM) with radial basis function kernel, so it could fit to data that was not linearly separable. Optimization of the pipeline was carried out with a grid search cross validation on the training split, in order to prevent overfitting and encourage good model generalization. In Chapter 5, we then implemented a deep neural network with integrated regularization layers that improved on the SVM in both consistency and performance.

Throughout this thesis, we developed the *subpred* Python packages, which allow for easy access to these methods through an API. Usage of this API is exemplified in Jupyter notebooks that are provided in the respective repositories. The *subpred-dl* package contains methods for creating protein datasets from a set of GO terms, for turning these proteins into ML features with encodings derived from sequence, 3D structure, evolutionary information and protein language models, and for training either the SVM-RBF pipeline with feature selection, or the DNN model on that dataset. The resulting ML model can then be used to predict the substrates of additional transmembrane transporters, either by assigning them into pre-defined substrate classes, or by finding all transporters of a particular substrate in a one-vs-rest approach against a background set. To ensure reproducibility and to speed up batch processing, all computation was done locally. The code automatically sets up local BLAST databases from

Uniref50 and Uniref90, downloads transformer models from Huggingface, makes use of CUDA if available, and creates its own sub-ontologies and GO-annotation subsets by filtering and extending the Gene Ontology and UniProtKB-GOA datasets. The package also makes extensive use of caching, so that time-consuming function calls, such as PSSM calculation or inference of language embeddings, only need to be executed once. We implemented several Python wrappers that allow for easy access to command line programs (CD-HIT, Fold-Seek, PSIBLAST), R packages (Biostrings, GoSemSim), and pre-trained protein language models (ProtT5, ProST5). They provide additional functions that are useful for the prediction of substrates and GO terms. The *subpred4* variant contains further methods for analyzing datasets, e.g. for creating hierarchical plots of ontologies where edges and nodes can be annotated with data from various sources, as well as the greedy optimization algorithm that reduced the redundancy in GO annotation sets by more than 90% in our tests. It also allows for the computation of all similarity measures and pairwise evaluation scores described in Chapter 4.

As the quantity and quality of input data is so important for training ML models for substrate prediction, the respective models will improve over time, as more experimental studies are carried out, and more of that information is integrated and organized into structured datasets, both automatically and through manual efforts.

One such effort was recently undertaken by the Resolute project [27], where the majority of the 464 human solute carrier (SLC) proteins were manually annotated by a team of researchers, with substrates derived from published literature. This work also provides interesting insights into the relationships between sequence identity, shared structural folds and substrate specificity of membrane transporters in the SLC super-family. As many of these proteins are now annotated with specific molecules that can be converted into molecular fingerprints, the idea of automatically clustering transport proteins through the chemical similarity of their substrates, which we introduced in Chapter 4, could now be applicable. Furthermore, the high quality of the dataset opens up the possibility of training ML models on this data, with the goal of finding homologues in other organisms, where substrate annotations are often missing. As the annotation effort also included known pharmaceutical compounds, the resulting models could provide interesting information to pharmaceutical research as well, e.g. about drug resistance or antibiotics.

Another development that will lead to improvements in data availability is the standardization of data storage, allowing for links and mappings between databases. Resources and tools such as GOA [288], InterProScan [295] or InterPro2GO [450] help to annotate sequence data with results from scientific studies, while projects that are part of the OBO Foundry [280] organize the available information into standardized and cross-compatible ontologies. These connections also make the data more accessible to machines, enabling automated reasoning with tools such as Protégé [286] and Robot [284]. In the future, specialized language models trained on this organized data could provide researchers with new hypotheses to test.

Further improvements to the accuracy of membrane transporter substrate prediction can also be achieved through advancements in machine learning algo-

rithms. In Chapter 5, we have shown that protein encodings derived with deep learning methods, paired with a regularized feedforward neural network (FNN) architecture, were able to beat previous approaches that relied on features derived from sequence and evolutionary information. More complex models and protein encodings, such as specialized sequence embeddings or different features derived from protein structure, have already been applied to general GO term prediction, with promising results [41][351]. Other methods add data from additional sources, such as annotated protein-protein interaction networks [451][427], or the hierarchy of “is_a” relations in the Gene Ontology [427][414] (see Section 2.2.1.5). While evaluation results from competitions show that the task of predicting all GO annotations of a protein is far from being solved [41], the winning methods could still provide improvements and new ideas for more specialized tasks, such as membrane transporter substrate prediction.

We also have some additional ideas that could improve the existing methods. These provide a good starting point for building on this work in the future. First, the ML features that delivered the best results in Chapter 3 and 4 were derived from PSSMs, by calculating the average scores for each amino acid type along the sequence. Wang et al. [452] lists a total of 21 methods for PSSM feature encoding, some of which could lead to improved performance of the resulting models. Similarly, we only calculated sequence embeddings with the ProtT5 and ProstT5 protein language models (pLM), the former of which had achieved the highest evaluation scores in comparative benchmarks. Other models that can be used for protein feature extraction, such as ESM-2 [410] or the other models that were introduced in the same study as ProtT5 [378], could potentially deliver better results for our specific use-case. Whereas we use embeddings from pLMs as input for simpler models, an alternative would be to fine-tune the pLM directly, turning it into a transporter substrate classifier. However, this would likely require significant computational resources and training time.

As many substrate classes and GO terms are only associated with few proteins, techniques from the field of few-shot learning could also be explored in the future. On one hand this encompasses transfer learning, where a DNN would be pre-trained on labeled or unlabeled data, in order to optimize the initial weights. This can improve both training time and classification performance, by ensuring that the model pays attention to the most useful input nodes immediately. On the other hand there are meta-learning approaches, where models “learn to learn” on classes with a very small number of samples, as is often the case for protein sets associated with GO terms. Metric-based meta-learners are especially interesting, since they follow a similar idea as the GO term clustering based on similarity scores between proteins that we tested in Chapter 4. One example are prototype networks [453], where a neural network iteratively learns to create optimized embeddings for the proteins, while a pairwise distance measure between these embeddings, such as the cosine distance, defines a metric space. The goal is to create embeddings that separate the classes (e.g. protein sets defined by GO terms) from each other in the space defined by this metric, while putting members of the same class close together. For every class, a *prototype* is calculated as the average embedding of all members. New samples can then be converted into the same embedding space, and the cosine distances between the embedded sample and the class prototypes determine the classification, in a “nearest neighbor” approach. Relational Networks (RNs) [454]

follow a similar idea. The two main differences are that they do not calculate prototypes, relying on pair-wise similarity scores instead, and that they learn both the embedding function and the distance metric through an iterative optimization process, instead of using the fixed cosine distance metric. This allows them to adapt to more complex datasets, although the computational cost is also increased.

Throughout this project, we also tried to implement classifiers that search for transporters of a particular substrate in a large set of proteins, using a one-vs-rest approach. However, we always had problems with the performance on the minority class, even when optimizing class weights, upsampling with SMOTE [352], or employing specialized evaluation scores such as focal loss. As few-shot methods are optimized for adapting to classes with a small number of samples, they could be a possible solution to this problem. For substrate prediction in general, calculating a prototype for each class would reduce the impact of class imbalance on the results, whereas the learned similarity score of RNs might be interesting for creating a model that adapts to the vast biological complexity found in multi-class membrane transporter datasets. These ideas, put together, could be the starting point for a more complex model architecture that would then be trained on the transporter datasets and protein encodings from Chapters 3 and 5, to find an optimal subset of functional annotations as in Chapter 4, and then assign new proteins with unknown functions to these classes with high accuracy.

In this thesis, we explored datasets of transmembrane transporters and created pipelines that find and label relevant proteins according to their substrate classes. We developed and expanded on feature encodings based on sequence, 3D structure, evolutionary information and state-of-the-art deep learning methods. Finally, we put these together with machine learning algorithms, such as support vector machines or deep neural networks, into full pipelines for data assembly, training and prediction, and made the implementation available as software packages. Building on this work, future advancements in specialized deep learning models, in hand-crafted protein encodings, along with manually curated training datasets, will enable the creation of machine learning pipelines that can annotate all transport proteins in any genome with their respective substrate classes and molecules, or find all transporters of a particular substrate in a given proteome. This tool will provide valuable information to various fields of research, including the design of specialized micro-organisms for biotechnological applications, the identification of drug exporters that contribute to resistance against pharmaceutical compounds, as well as basic research efforts that aim to understand the functions of all genes and proteins in any species.

Bibliography

- [1] Bruce Alberts, Rebecca Heald, Alexander Johnson, David Morgan, and Martin Raff. Protein function. In *Molecular Biology of the Cell 7th Edition*. Norton & Company, 2022.
- [2] Wilfred D Stein and Thomas Litman. *Channels, carriers, and pumps: an introduction to membrane transport*. Elsevier, 2014.
- [3] Nicole J Yang and Marlon J Hinner. Getting across the cell membrane: an overview for small molecules, peptides, and proteins. *Site-Specific Protein Labeling: Methods and Protocols*, pages 29–53, 2015.
- [4] Douglas B Kell and Stephen G Oliver. How drugs get into cells: tested and testable predictions to help discriminate between transporter-mediated uptake and lipoidal bilayer diffusion. *Frontiers in pharmacology*, 5:113250, 2014.
- [5] Atin Sharma, Vivek Kumar Gupta, and Ranjana Pathania. Efflux pump inhibitors for bacterial pathogens: From bench to bedside. *Indian Journal of Medical Research*, 149(2):129–145, 2019.
- [6] Karol Bukowski, Mateusz Kciuk, and Renata Kontek. Mechanisms of multidrug resistance in cancer chemotherapy. *International journal of molecular sciences*, 21(9):3233, 2020.
- [7] John P Overington, Bissan Al-Lazikani, and Andrew L Hopkins. How many drug targets are there? *Nature reviews Drug discovery*, 5(12):993–996, 2006.
- [8] Swagatika Sahoo, Maike K Aurich, Jon J Jonsson, and Ines Thiele. Membrane transporters in a human genome-scale metabolic knowledgebase and their implications for disease. *Frontiers in physiology*, 5:55648, 2014.
- [9] Matthew A Oberhardt, Bernhard Ø Palsson, and Jason A Papin. Applications of genome-scale metabolic reconstructions. *Molecular systems biology*, 5(1):320, 2009.
- [10] Douglas McCloskey, Bernhard Ø Palsson, and Adam M Feist. Basic and applied uses of genome-scale metabolic network reconstructions of escherichia coli. *Molecular systems biology*, 9(1):661, 2013.
- [11] Jennifer Dumont, Don Euwart, Baisong Mei, Scott Estes, and Rashmi Kshirsagar. Human cell lines for biopharmaceutical manufacturing: his-

- tory, status, and future perspectives. *Critical reviews in biotechnology*, 36(6):1110–1122, 2016.
- [12] Diethard Mattanovich, Michael Sauer, and Brigitte Gasser. Yeast biotechnology: teaching the old dog new tricks. *Microbial cell factories*, 13:1–5, 2014.
- [13] XQ Zhao and FW Bai. Yeast flocculation: new story in fuel ethanol production. *Biotechnology advances*, 27(6):849–856, 2009.
- [14] Tianxun Zhou, Rinta Reji, Ryanjit Singh Kairon, and Keng Hwee Chiam. A review of algorithmic approaches for cell culture media optimization. *Frontiers in Bioengineering and Biotechnology*, 11:1195294, 2023.
- [15] Takamasa Hashizume, Yuki Ozawa, and Bei-Wen Ying. Employing active learning in the optimization of culture medium for mammalian cells. *npj systems biology and applications*, 9(1):20, 2023.
- [16] Steven A van der Hoek and Irina Borodina. Transporter engineering in microbial cell factories: the ins, the outs, and the in-betweens. *Current opinion in biotechnology*, 66:186–194, 2020.
- [17] Florian F Bauer, Patrick Govender, and Michael C Bester. Yeast flocculation and its biotechnological relevance. *Applied microbiology and biotechnology*, 88:31–39, 2010.
- [18] Bradley W Wright, Zixin Yi, Jonathan S Weissman, and Jin Chen. The dark proteome: translation from noncanonical open reading frames. *Trends in Cell Biology*, 32(3):243–258, 2022.
- [19] Adrián César-Razquin, Berend Snijder, Tristan Frappier-Brinton, Ruth Isserlin, Gergely Gyimesi, Xiaoyun Bai, Reinhart A Reithmeier, David Hepworth, Matthias A Hediger, Aled M Edwards, et al. A call for systematic research on solute carriers. *Cell*, 162(3):478–487, 2015.
- [20] Giulio Superti-Furga, Daniel Lackner, Tabea Wiedmer, Alvaro Ingles-Prieto, Barbara Barbosa, Enrico Girardi, Ulrich Goldmann, Bettina Gürtl, Kristaps Klavins, Christoph Klimek, et al. The resolute consortium: unlocking slc transporters for drug discovery. *Nature reviews Drug discovery*, 19(7):429–430, 2020.
- [21] John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, et al. Highly accurate protein structure prediction with alphafold. *Nature*, 596(7873):583–589, 2021.
- [22] Hajar Owji, Navid Nezafat, Manica Negahdaripour, Ali Hajiebrahimi, and Younes Ghasemi. A comprehensive review of signal peptides: Structure, roles, and applications. *European journal of cell biology*, 97(6):422–441, 2018.
- [23] Felix Teufel, José Juan Almagro Armenteros, Alexander Rosenberg Johansen, Magnús Halldór Gíslason, Silas Irby Pihl, Konstantinos D Tsirigos, Ole Winther, Søren Brunak, Gunnar von Heijne, and Henrik Nielsen. Signalp 6.0 predicts all five types of signal peptides using protein language models. *Nature biotechnology*, 40(7):1023–1025, 2022.

- [24] Stephen F Altschul, Warren Gish, Webb Miller, Eugene W Myers, and David J Lipman. Basic local alignment search tool. *Journal of molecular biology*, 215(3):403–410, 1990.
- [25] L Steven Johnson, Sean R Eddy, and Elon Portugaly. Hidden markov model speed heuristic and iterative hmm search procedure. *BMC bioinformatics*, 11(1):431, 2010.
- [26] Liisa B Koski and G Brian Golding. The closest blast hit is often not the nearest neighbor. *Journal of molecular evolution*, 52(6):540–542, 2001.
- [27] Ulrich Goldmann, Tabea Wiedmer, Andrea Garofoli, Vitaly Sedlyarov, Manuel Bichler, Ben Haladik, Gernot Wolf, Eirini Christodoulaki, Alvaro Ingles-Prieto, Evandro Ferrada, et al. Data-and knowledge-derived functional landscape of human solute carriers. *Molecular Systems Biology*, 21(6):599–631, 2025.
- [28] David Drew and Olga Boudker. Shared molecular mechanisms of membrane transporters. *Annual review of biochemistry*, 85:543–572, 2016.
- [29] David Drew, Rachel A North, Kumar Nagarathinam, and Mikio Tanabe. Structures and general transport mechanisms by the major facilitator superfamily (mfs). *Chemical reviews*, 121(9):5289–5335, 2021.
- [30] Kelvin Wong, Jerome Ma, Alice Rothnie, Philip C Biggin, and Ian D Kerr. Towards understanding promiscuity in multidrug efflux pumps. *Trends in biochemical sciences*, 39(1):8–16, 2014.
- [31] Yong Yin, Xiao He, Paul Szewczyk, That Nguyen, and Geoffrey Chang. Structure of the multidrug transporter emrd from escherichia coli. *Science*, 312(5774):741–744, 2006.
- [32] Devanshu Kurre, Phuoc X Dang, Le TM Le, Varun V Gadkari, and Amer Alam. Structural insights into binding-site access and ligand recognition by human abcb1. *The EMBO Journal*, pages 1–16, 2025.
- [33] Benoît Roux. Ion channels and ion selectivity. *Essays in biochemistry*, 61(2):201–209, 2017.
- [34] Christoph Thomas and Robert Tampé. Structural and mechanistic principles of abc transporters. *Annual review of biochemistry*, 89(1):605–636, 2020.
- [35] Yufeng Zhou, Joao H Morais-Cabral, Amelia Kaufman, and Roderick MacKinnon. Chemistry of ion coordination and hydration revealed by a k⁺ channel–fab complex at 2.0 Å resolution. *Nature*, 414(6859):43–48, 2001.
- [36] Todor Dudev and Carmay Lim. Ion selectivity strategies of sodium channel selectivity filters. *Accounts of chemical research*, 47(12):3580–3587, 2014.
- [37] Abraham B Chang, Ron Lin, W Keith Studley, Can V Tran, and Milton H Saier, Jr. Phylogeny as a guide to structure and function of membrane transport proteins. *Molecular membrane biology*, 21(3):171–181, 2004.

- [38] Ahmad Barghash and Volkhard Helms. Transferring functional annotations of membrane transporters on the basis of sequence similarity and sequence motifs. *BMC bioinformatics*, 14(1):343, 2013.
- [39] Nadine S Schaadt, Jan Christoph, and Volkhard Helms. Classifying substrate specificities of membrane transporters from *arabidopsis thaliana*. *Journal of chemical information and modeling*, 50(10):1899–1905, 2010.
- [40] Nitish K Mishra, Junil Chang, and Patrick X Zhao. Prediction of membrane transport proteins and their substrate specificities using primary sequence information. *PLoS One*, 9(6):e100278, 2014.
- [41] Naihui Zhou, Yuxiang Jiang, Timothy R Bergquist, Alexandra J Lee, Balint Z Kacsoh, Alex W Crocker, Kimberley A Lewis, George Georgiou, Huy N Nguyen, Md Nafiz Hamid, et al. The cafa challenge reports improved protein function prediction and new functional annotations for hundreds of genes through experimental screens. *Genome biology*, 20:1–23, 2019.
- [42] Michel Van Kempen, Stephanie S Kim, Charlotte Tumescheit, Milot Mirdita, Jeongjae Lee, Cameron LM Gilchrist, Johannes Söding, and Martin Steinegger. Fast and accurate protein structure search with foldseek. *Nature biotechnology*, 42(2):243–246, 2024.
- [43] Matthew E Ritchie, Belinda Phipson, DI Wu, Yifang Hu, Charity W Law, Wei Shi, and Gordon K Smyth. limma powers differential expression analyses for rna-sequencing and microarray studies. *Nucleic acids research*, 43(7):e47–e47, 2015.
- [44] Hadley Wickham, Mara Averick, Jennifer Bryan, Winston Chang, Lucy D’Agostino McGowan, Romain François, Garrett Grolemond, Alex Hayes, Lionel Henry, Jim Hester, et al. Welcome to the tidyverse. *Journal of open source software*, 4(43):1686, 2019.
- [45] Wolfgang Huber, Vincent J Carey, Robert Gentleman, Simon Anders, Marc Carlson, Benilton S Carvalho, Hector Corrada Bravo, Sean Davis, Laurent Gatto, Thomas Girke, et al. Orchestrating high-throughput genomic analysis with bioconductor. *Nature methods*, 12(2):115–121, 2015.
- [46] Yoav Benjamini and Yosef Hochberg. Controlling the false discovery rate: a practical and powerful approach to multiple testing. *Journal of the Royal statistical society: series B (Methodological)*, 57(1):289–300, 1995.
- [47] GO Consortium. The gene ontology resource: 20 years and still going strong. *Nucleic Acids Res*, 47(D1):D330–D338, 2019.
- [48] Minoru Kanehisa, Miho Furumichi, Mao Tanabe, Yoko Sato, and Kanae Morishima. Kegg: new perspectives on genomes, pathways, diseases and drugs. *Nucleic acids research*, 45(D1):D353–D361, 2017.
- [49] Ege Ulgen, Ozan Ozisik, and Osman Ugur Sezerman. pathfinder: an r package for comprehensive identification of enriched pathways in omics data through active subnetworks. *Frontiers in genetics*, 10:858, 2019.
- [50] Awais Athar, Anja Füllgrabe, Nancy George, Haider Iqbal, Laura Huerta, Ahmed Ali, Catherine Snow, Nuno A Fonseca, Robert Petryszak, Irene

- Papatheodorou, et al. Arrayexpress update—from bulk to single-cell expression data. *Nucleic Acids Res.*, 47(D1):D711–D715, 2019.
- [51] Michael L Waskom. Seaborn: statistical data visualization. *Journal of Open Source Software*, 6(60):3021, 2021.
- [52] John D Hunter. Matplotlib: A 2d graphics environment. *Computing in science & engineering*, 9(03):90–95, 2007.
- [53] Hadley Wickham and Carson Sievert. *ggplot2: elegant graphics for data analysis*, volume 10. springer New York, 2009.
- [54] Gregory R Warnes, Ben Bolker, Lodewijk Bonebakker, Robert Gentleman, Wolfgang Huber, Andy Liaw, Thomas Lumley, Martin Maechler, Arni Magnusson, Steffen Moeller, et al. *gplots: Various r programming tools for plotting data. (No Title)*, 2005.
- [55] Willy Hugo, Jesse M Zaretsky, LU Sun, Chunying Song, Blanca Homet Moreno, Siwen Hu-Lieskovan, Beata Berent-Maoz, Jia Pang, Bartosz Chmielowski, Grace Cherry, et al. Genomic and transcriptomic features of response to anti-pd-1 therapy in metastatic melanoma. *Cell*, 165(1):35–44, 2016.
- [56] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.
- [57] Bruce Alberts, Rebecca Heald, Alexander Johnson, David Morgan, and Martin Raff. *Molecular Biology of the Cell 7th Edition*. Norton & Company, 2022.
- [58] Carl Ivar Branden and John Tooze. *Introduction to protein structure*. Garland Science, 2012.
- [59] Tomasz Fekner and Michael K Chan. The pyrrolysine translational machinery as a genetic-code expansion tool. *Current opinion in chemical biology*, 15(3):387–391, 2011.
- [60] Linus Pauling, Robert B Corey, and Herman R Branson. The structure of proteins: two hydrogen-bonded helical configurations of the polypeptide chain. *Proceedings of the National Academy of Sciences*, 37(4):205–211, 1951.
- [61] Peter D Sun, Christine E Foster, and Jeffrey C Boyington. Overview of protein structural and functional folds. *Current protocols in protein science*, 35(1):17–1, 2004.
- [62] Richard B Cooley, Daniel J Arp, and P Andrew Karplus. Evolutionary origin of a secondary structure: π -helices as cryptic but widespread insertional variations of α -helices that enhance protein functionality. *Journal of molecular biology*, 404(2):232–246, 2010.

- [63] Ricardo Simão Vieira-Pires and João Henrique Morais-Cabral. 310 helices in channels and other membrane proteins. *Journal of General Physiology*, 136(6):585–592, 2010.
- [64] Rakesh Trivedi and Hampapathalu Adimurthy Nagarajaram. Intrinsically disordered proteins: an overview. *International journal of molecular sciences*, 23(22):14050, 2022.
- [65] Bruce Alberts, Alexander Johnson, Julian Lewis, Martin Raff, Keith Roberts, and Peter Walter. The shape and structure of proteins. In *Molecular Biology of the Cell. 4th edition*. Garland Science, 2002.
- [66] Nataraj S Pagadala, Khajamohiddin Syed, and Jack Tuszynski. Software for molecular docking: a review. *Biophysical reviews*, 9:91–102, 2017.
- [67] John A Capra and Mona Singh. Predicting functionally important residues from sequence conservation. *Bioinformatics*, 23(15):1875–1882, 2007.
- [68] Steven Henikoff and Jorja G Henikoff. Amino acid substitution matrices from protein blocks. *Proceedings of the National Academy of Sciences*, 89(22):10915–10919, 1992.
- [69] Fabian Sievers, Andreas Wilm, David Dineen, Toby J Gibson, Kevin Karplus, Weizhong Li, Rodrigo Lopez, Hamish McWilliam, Michael Remmert, Johannes Söding, et al. Fast, scalable generation of high-quality protein multiple sequence alignments using clustal omega. *Molecular systems biology*, 7(1):539, 2011.
- [70] Peer Bork and Eugene V Koonin. Protein sequence motifs. *Current opinion in structural biology*, 6(3):366–376, 1996.
- [71] International Human Genome Sequencing Consortium et al. Initial sequencing and analysis of the human genome. *Nature*, 409(6822):860, 2001.
- [72] Robert D Finn, Alex Bateman, Jody Clements, Penelope Coghill, Ruth Y Eberhardt, Sean R Eddy, Andreas Heger, Kirstie Hetherington, Liisa Holm, Jaina Mistry, et al. Pfam: the protein families database. *Nucleic acids research*, 42(D1):D222–D230, 2013.
- [73] Antonia Stank, Daria B Kokh, Jonathan C Fuller, and Rebecca C Wade. Protein binding pocket dynamics. *Accounts of chemical research*, 49(5):809–815, 2016.
- [74] Vivian Osei Poku and Surtaj Hussain Iram. A critical review on modulators of multidrug resistance protein 1 in cancer cells. *PeerJ*, 10:e12594, 2022.
- [75] Andrew L Hopkins and Colin R Groom. The druggable genome. *Nature reviews Drug discovery*, 1(9):727–730, 2002.
- [76] Antonio Lavecchia and Carmen Di Giovanni. Virtual screening strategies in drug discovery: a critical review. *Current medicinal chemistry*, 20(23):2839–2860, 2013.
- [77] Wesley E Stites. Protein- protein interactions: interface structure, binding thermodynamics, and mutational analysis. *Chemical reviews*, 97(5):1233–1250, 1997.

- [78] Li C Xue, Drena Dobbs, Alexandre MJJ Bonvin, and Vasant Honavar. Computational prediction of protein interfaces: A review of data driven methods. *FEBS letters*, 589(23):3516–3526, 2015.
- [79] Peter S Shen, Xiaoyong Yang, Paul G DeCaen, Xiaowen Liu, David Bulkley, David E Clapham, and Erhu Cao. The structure of the polycystic kidney disease channel pkd2 in lipid nanodiscs. *Cell*, 167(3):763–773, 2016.
- [80] Nicholas MI Taylor, Ioannis Manolaridis, Scott M Jackson, Julia Kowal, Henning Stahlberg, and Kaspar P Locher. Structure of the human multidrug transporter abcg2. *Nature*, 546(7659):504–509, 2017.
- [81] S Jonathan Singer and Garth L Nicolson. The fluid mosaic model of the structure of cell membranes. *Science*, 175(4023):720–731, 1972.
- [82] Félix M Goñi. The basic structure and dynamics of cell membranes: An update of the singer–nicolson model. *Biochimica et Biophysica Acta (BBA)-Biomembranes*, 1838(6):1467–1476, 2014.
- [83] Takeshi Harayama and Howard Riezman. Understanding the diversity of membrane lipid composition. *Nature reviews Molecular cell biology*, 19(5):281–296, 2018.
- [84] Garth L Nicolson. Update of the 1972 singer-nicolson fluid-mosaic model of membrane structure. *Discoveries*, 1(1), 2013.
- [85] Bartosz Kowalik, Thomas Schubert, Hirofumi Wada, Motomu Tanaka, Roland R Netz, and Emanuel Schneck. Combination of md simulations with two-state kinetic rate modeling elucidates the chain melting transition of phospholipid bilayers for different hydration levels. *The Journal of Physical Chemistry B*, 119(44):14157–14167, 2015.
- [86] Liping Sun and Rainer A Böckmann. Membrane phase transition during heating and cooling: molecular insight into reversible melting. *European Biophysics Journal*, 47(2):151–164, 2018.
- [87] Leonhard Möckl. The emerging role of the mammalian glycocalyx in functional membrane organization and immune system regulation. *Frontiers in cell and developmental biology*, 8:253, 2020.
- [88] Michael A McGuckin, Sara K Lindén, Philip Sutton, and Timothy H Florin. Mucin dynamics and enteric pathogens. *Nature Reviews Microbiology*, 9(4):265–278, 2011.
- [89] Brian K Brandley and Ronald L Schnaar. Cell-surface carbohydrates in cell recognition and response. *Journal of leukocyte biology*, 40(1):97–111, 1986.
- [90] S Bafna, Sukwinder Kaur, and Surinder Kumar Batra. Membrane-bound mucins: the mechanistic basis for alterations in the growth and survival of cancer cells. *Oncogene*, 29(20):2893–2904, 2010.
- [91] Nathan Sharon and Halina Lis. Microbial lectins and their glycoprotein receptors. In *New Comprehensive Biochemistry*, volume 29, pages 475–506. Elsevier, 1997.

- [92] Michael P Sheetz, Julia E Sable, and Hans-Günther Döbereiner. Continuous membrane-cytoskeleton adhesion requires continuous accommodation to lipid and cytoskeleton dynamics. *Annu. Rev. Biophys. Biomol. Struct.*, 35(1):417–434, 2006.
- [93] Catherine D Strader, Tung Ming Fong, Michael P Grazlano, and Michael R Tota. The family of g-protein-coupled receptors. *The FASEB journal*, 9(9):745–754, 1995.
- [94] Deborah M Boes, Albert Godoy-Hernandez, and Duncan GG McMillan. Peripheral membrane proteins: promising therapeutic targets across domains of life. *Membranes*, 11(5):346, 2021.
- [95] Taroh Kinoshita. Glycosylphosphatidylinositol (gpi) anchors: biochemistry and cell biology: introduction to a thematic review series. *Journal of lipid research*, 57(1):4–5, 2016.
- [96] Reinhard Schweitzer-Stenner. Relating the multi-functionality of cytochrome c to membrane binding and structural conversion. *Biophysical Reviews*, 10(4):1151–1185, 2018.
- [97] Ronald KH Liem. Cytoskeletal integrators: the spectrin superfamily. *Cold Spring Harbor Perspectives in Biology*, 8(10):a018259, 2016.
- [98] Kutti R Vinothkumar and Richard Henderson. Structures of membrane proteins. *Quarterly reviews of Biophysics*, 43(1):65–158, 2010.
- [99] Anders Krogh, BjoÈrn Larsson, Gunnar Von Heijne, and Erik LL Sonnhammer. Predicting transmembrane protein topology with a hidden markov model: application to complete genomes. *Journal of molecular biology*, 305(3):567–580, 2001.
- [100] Renhong Yan, Xin Zhao, Jianlin Lei, and Qiang Zhou. Structure of the human lat1–4f2hc heteromeric amino acid transporter complex. *Nature*, 568(7750):127–130, 2019.
- [101] Peilong Lu, Dan Ma, Chuangye Yan, Xinqi Gong, Mingjian Du, and Yigong Shi. Structure and mechanism of a eukaryotic transmembrane ascorbate-dependent oxidoreductase. *Proceedings of the National Academy of Sciences*, 111(5):1813–1818, 2014.
- [102] Helen Watson. Biological membranes. *Essays in biochemistry*, 59:43–69, 2015.
- [103] Martin Beck and Ed Hurt. The nuclear pore complex: understanding its function through structural insight. *Nature reviews Molecular cell biology*, 18(2):73–89, 2017.
- [104] Shuliang Chen, Peter Novick, and Susan Ferro-Novick. Er structure and function. *Current opinion in cell biology*, 25(4):428–433, 2013.
- [105] Kornelius Zeth. Structure and evolution of mitochondrial outer membrane proteins of β -barrel topology. *Biochimica Et Biophysica Acta (BBA)-Bioenergetics*, 1797(6-7):1292–1299, 2010.

- [106] Ru-Zhou Zhao, Shuai Jiang, Lin Zhang, and Zhi-Bin Yu. Mitochondrial electron transport chain, ros generation and uncoupling. *International journal of molecular medicine*, 44(1):3–15, 2019.
- [107] Carmen A Mannella. Structure and dynamics of the mitochondrial inner membrane cristae. *Biochimica et Biophysica Acta (BBA)-Molecular Cell Research*, 1763(5-6):542–548, 2006.
- [108] Karolina Boguszezewska, Michał Szewczuk, Julia Kaźmierczak-Barańska, and Bolesław T Karwowski. The similarities between human mitochondria and bacteria in the context of structure, genome, and base excision repair system. *Molecules*, 25(12):2857, 2020.
- [109] James W Fairman, Nicholas Noinaj, and Susan K Buchanan. The structural biology of β -barrel membrane proteins: a summary of recent reports. *Current opinion in structural biology*, 21(4):523–531, 2011.
- [110] Ruairidh Edwards, Ross Eaglesfield, and Kostas Tokatlidis. The mitochondrial intermembrane space: the most constricted mitochondrial sub-compartment with the largest variety of protein import pathways. *Open biology*, 11(3):210002, 2021.
- [111] Samuel I Miller and Nina R Salama. The gram-negative bacterial periplasm: Size matters. *PLoS biology*, 16(1):e2004935, 2018.
- [112] Bing K Jap and Peter J Walian. Biophysics of the structure and function of porins. *Quarterly reviews of biophysics*, 23(4):367–403, 1990.
- [113] Qingshan Luo, Xu Yang, Shan Yu, Huigang Shi, Kun Wang, Le Xiao, Guangyu Zhu, Chuanqi Sun, Tingting Li, Dianfan Li, et al. Structural basis for lipopolysaccharide extraction by abc transporter lptb2fg. *Nature structural & molecular biology*, 24(5):469–474, 2017.
- [114] Shambhavi Garde, Pavan Kumar Chodiseti, and Manjula Reddy. Peptidoglycan: structure, synthesis, and regulation. *EcoSal Plus*, 9(2), 2021.
- [115] Sean D Conner and Sandra L Schmid. Regulated portals of entry into the cell. *Nature*, 422(6927):37–44, 2003.
- [116] Jatta Huotari and Ari Helenius. Endosome maturation. *The EMBO journal*, 30(17):3481–3500, 2011.
- [117] Peter R Shepherd and Barbara B Kahn. Glucose transporters and insulin action—implications for insulin resistance and diabetes mellitus. *New England journal of medicine*, 341(4):248–257, 1999.
- [118] Milton H Saier Jr, Can V Tran, and Ravi D Barabote. Tcdb: the transporter classification database for membrane transport protein analyses and information. *Nucleic acids research*, 34(suppl_1):D181–D186, 2006.
- [119] Stephan Wilkens. Structure and mechanism of abc transporters. *F1000prime reports*, 7, 2015.
- [120] R Astashkin, K Kovalev, S Bukhdruker, S Vaganova, A Kuzmin, A Alekseev, T Balandin, D Zabelskii, I Gushchin, A Royant, et al. Structural insights into light-driven anion pumping in cyanobacteria. *Nature Communications*, 13(1):6460, 2022.

- [121] RJW Brooijmans, B Poolman, GK Schuurman-Wolters, WM De Vos, and J Hugenholtz. Generation of a membrane potential by *lactococcus lactis* through aerobic electron transport. *Journal of Bacteriology*, 189(14):5203–5209, 2007.
- [122] Robert C Ford and Konstantinos Beis. Learning the abcs one at a time: structure and mechanism of abc transporters. *Biochemical Society Transactions*, 47(1):23–36, 2019.
- [123] Anthony M George and Peter M Jones. Perspectives on the structure–function of abc transporters: the switch and constant contact models. *Progress in biophysics and molecular biology*, 109(3):95–107, 2012.
- [124] Christoph Thomas and Robert Tampé. Multifaceted structures and mechanisms of abc transport systems in health and disease. *Current opinion in structural biology*, 51:116–128, 2018.
- [125] Kaspar P Locher. Mechanistic diversity in atp-binding cassette (abc) transporters. *Nature structural & molecular biology*, 23(6):487–493, 2016.
- [126] Ronnie P-A Berntsson, Sander HJ Smits, Lutz Schmitt, Dirk-Jan Slotboom, and Bert Poolman. A structural classification of substrate-binding proteins. *FEBS letters*, 584(12):2606–2617, 2010.
- [127] Vladimir M Korkhov, Samantha A Mireku, and Kaspar P Locher. Structure of amp-pnp-bound vitamin b12 transporter btucd-f. *Nature*, 490(7420):367–372, 2012.
- [128] Michael L Oldham, Dheeraj Khare, Florante A Quirocho, Amy L Davidson, and Jue Chen. Crystal structure of a catalytic intermediate of the maltose transporter. *Nature*, 450(7169):515–521, 2007.
- [129] David Sehnal, Sebastian Bittrich, Mandar Deshpande, Radka Svobodová, Karel Berka, Václav Bazgier, Sameer Velankar, Stephen K Burley, Jaroslav Koča, and Alexander S Rose. Mol*viewer: modern web app for 3d visualization and analysis of large biomolecular structures. *Nucleic acids research*, 49(W1):W431–W437, 2021.
- [130] Lotteke JYM Swier, Albert Guskov, and Dirk J Slotboom. Structural insight in the toppling mechanism of an energy-coupling factor transporter. *Nature Communications*, 7(1):11072, 2016.
- [131] Nathan K Karpowich, Jin Mei Song, Nicolette Cocco, and Da-Neng Wang. Atp binding drives substrate capture in an ecf transporter by a release-and-catch mechanism. *Nature structural & molecular biology*, 22(7):565–571, 2015.
- [132] S Rempel, WK Stanek, and DJ Slotboom. Ecf-type atp-binding cassette transporters. *Annual review of biochemistry*, 88(1):551–576, 2019.
- [133] Anna Seelig. P-glycoprotein: one mechanism, many tasks and the consequences for pharmacotherapy of cancers. *Frontiers in oncology*, 10:576559, 2020.
- [134] Cátia A Bonito, Ricardo J Ferreira, Maria-José U Ferreira, Jean-Pierre Gillet, M Natália DS Cordeiro, and Daniel JVA Dos Santos. Theoretical

- insights on helix repacking as the origin of p-glycoprotein promiscuity. *Scientific reports*, 10(1):9823, 2020.
- [135] Hang Zhang, Haiwei Xu, Charles R Ashby Jr, Yehuda G Assaraf, Zhe-Sheng Chen, and Hong-Min Liu. Chemical molecular-based approach to overcome multidrug resistance in cancer by targeting p-glycoprotein (p-gp). *Medicinal research reviews*, 41(1):525–555, 2021.
- [136] Frances J Sharom. The p-glycoprotein multidrug transporter. *Essays in biochemistry*, 50:161–178, 2011.
- [137] Frances J Sharom. Flipping and flopping—lipids on the move. *IUBMB life*, 63(9):736–746, 2011.
- [138] Wei Mi, Yanyan Li, Sung Hwan Yoon, Robert K Ernst, Thomas Walz, and Maofu Liao. Structural basis of msba-mediated lipopolysaccharide transport. *Nature*, 549(7671):233–237, 2017.
- [139] Scott M Jackson, Ioannis Manolaridis, Julia Kowal, Melanie Zechner, Nicholas MI Taylor, Manuel Bause, Stefanie Bauer, Ruben Bartholomaeus, Guenther Bernhardt, Burkhard Koenig, et al. Structural basis of small-molecule inhibition of human multidrug transporter abcg2. *Nature structural & molecular biology*, 25(4):333–340, 2018.
- [140] Allister Crow, Nicholas P Greene, Elise Kaplan, and Vassilis Koronakis. Structure and mechanotransmission mechanism of the macb abc transporter superfamily. *Proceedings of the National Academy of Sciences*, 114(47):12572–12577, 2017.
- [141] Anthony WP Fitzpatrick, Salomé Llabrés, Arthur Neuberger, James N Blaza, Xiao-chen Bai, Ui Okada, Satoshi Murakami, Hendrik W Van Veen, Ulrich Zachariae, Sjors HW Scheres, et al. Structure of the macab-tolc abc-type tripartite multidrug efflux pump. *Nature microbiology*, 2(7):1–8, 2017.
- [142] Michael Dean, Yannick Hamon, and Giovanna Chimini. The human atp-binding cassette (abc) transporter superfamily. *Journal of lipid research*, 42(7):1007–1017, 2001.
- [143] Amer Alam and Kaspar P Locher. Structure and mechanism of human abc transporters. *Annual Review of Biophysics*, 52(1):275–300, 2023.
- [144] VM Zubareva, AS Lapashina, TE Shugaeva, AV Litvin, and BA Feniouk. Rotary ion-translocating atpases/atp synthases: Diversity, similarities, and differences. *Biochemistry (Moscow)*, 85:1613–1630, 2020.
- [145] Hiroshi Ueno, Toshiharu Suzuki, Kazuhiko Kinoshita Jr, and Masasuke Yoshida. Atp-driven stepwise rotation of f_of₁-atp synthase. *Proceedings of the National Academy of Sciences*, 102(5):1333–1338, 2005.
- [146] Vanessa Leone, Denys Pogoryelov, Thomas Meier, and José D Faraldo-Gómez. On the principle of ion selectivity in na⁺/h⁺-coupled membrane proteins: experimental and theoretical studies of an atp synthase rotor. *Proceedings of the National Academy of Sciences*, 112(10):E1057–E1066, 2015.

- [147] Ryuta Kanai, Haruo Ogawa, Bente Vilsen, Flemming Cornelius, and Chikashi Toyoshima. Crystal structure of a Na^+ -bound Na^+ , K^+ -atpase preceding the e1p state. *Nature*, 502(7470):201–206, 2013.
- [148] Takehiro Shinoda, Haruo Ogawa, Flemming Cornelius, and Chikashi Toyoshima. Crystal structure of the sodium–potassium pump at 2.4 Å resolution. *Nature*, 459(7245):446–450, 2009.
- [149] Huan Rui, Pablo Artigas, and Benoît Roux. The selectivity of the Na^+/K^+ -pump is controlled by binding site protonation and self-correcting occlusion. *Elife*, 5:e16616, 2016.
- [150] Jesper V Møller, Birte Juul, and Marc le Maire. Structural organization, ion transport, and energy transduction of p-type atpases. *Biochimica et Biophysica Acta (BBA)-Reviews on Biomembranes*, 1286(1):1–51, 1996.
- [151] Henry Chan, Vartan Babayan, Elya Blyumin, Charmy Gandhi, Kunal Hak, Danielle Harake, Kris Kumar, Perry Lee, Tze T Li, Hao Yi Liu, et al. The p-type atpase superfamily. *Journal of molecular microbiology and biotechnology*, 19(1-2):5–104, 2010.
- [152] Jörg-Christian Greie. The kdpfabc complex from escherichia coli: a chimeric K^+ transporter merging ion pumps with ion channels. *European journal of cell biology*, 90(9):705–710, 2011.
- [153] AK Verma, AG Filoteo, DR Stanford, Eric D Wieben, JT Penniston, Emanuel E Strehler, R Fischer, R Heim, G Vogel, and S Mathews. Complete primary structure of a human plasma membrane Ca^{2+} pump. *Journal of Biological Chemistry*, 263(28):14152–14159, 1988.
- [154] Sivaram Chintalapati, Rana Al Kurdi, Anke C Terwisscha van Scheltinga, and Werner Kühlbrandt. Membrane structure of ctra3, a copper-transporting p-type-atpase from aquifex aeolicus. *Journal of molecular biology*, 378(3):581–595, 2008.
- [155] Nicolás Reyes and David C Gadsby. Ion permeation through the Na^+ , K^+ -atpase. *Nature*, 443(7110):470–474, 2006.
- [156] Lucy R Forrest and Gary Rudnick. The rocking bundle: a mechanism for ion-coupled solute flux by symmetrical transporters. *Physiology*, 24(6):377–386, 2009.
- [157] Oliver Beckstein and Fiona Naughton. General principles of secondary active transporter function. *Biophysics Reviews*, 3(1), 2022.
- [158] Matthias A Hediger, Michael F Romero, Ji-Bin Peng, Andreas Rolfs, Hitomi Takanaga, and Elspeth A Bruford. The abcs of solute carriers: physiological, pathological and therapeutic implications of human membrane transport proteins: Introduction. *Pflügers Archiv*, 447:465–468, 2004.
- [159] Lucy R Forrest, Reinhard Krämer, and Christine Ziegler. The structural basis of secondary active transport mechanisms. *Biochimica et Biophysica Acta (BBA)-Bioenergetics*, 1807(2):167–188, 2011.

- [160] Martin Klingenberg. Ligand- protein interaction in biomembrane carriers. the induced transition fit of transport catalysis. *Biochemistry*, 44(24): 8563–8570, 2005.
- [161] Christopher J Law, Peter C Maloney, and Da-Neng Wang. Ins and outs of major facilitator superfamily antiporters. *Annu. Rev. Microbiol.*, 62: 289–305, 2008.
- [162] Olga Boudker and Grégory Verdon. Structural perspectives on secondary active transporters. *Trends in pharmacological sciences*, 31(9):418–426, 2010.
- [163] Christos Gournas, Alexandros Athanasopoulos, and Vicky Sophianopoulou. On the evolution of specificity in members of the yeast amino acid transporter family as parts of specific metabolic pathways. *International journal of molecular sciences*, 19(5):1398, 2018.
- [164] Yigong Shi. Common folds and transport mechanisms of secondary active transporters. *Annual review of biophysics*, 42(1):51–72, 2013.
- [165] Juke S Lolkema and Dirk-Jan Slotboom. The major amino acid transporter superfamily has a similar core structure as na⁺-galactose and na⁺-leucine transporters. *Molecular membrane biology*, 25(6-7):567–570, 2008.
- [166] Dong Deng, Pengcheng Sun, Chuangye Yan, Meng Ke, Xin Jiang, Lei Xiong, Wenlin Ren, Kunio Hirata, Masaki Yamamoto, Shilong Fan, et al. Molecular basis of ligand recognition and transport by glucose transporters. *Nature*, 526(7573):391–396, 2015.
- [167] Mike Mueckler and Bernard Thorens. The slc2 (glut) family of membrane transporters. *Molecular aspects of medicine*, 34(2-3):121–138, 2013.
- [168] Dong Deng and Nieng Yan. Glut, sgl, and sweet: Structural and mechanistic investigations of the glucose transporters. *Protein Science*, 25(3): 546–558, 2016.
- [169] Vamsee S Reddy, Maksim A Shlykov, Rostislav Castillo, Eric I Sun, and Milton H Saier Jr. The major facilitator superfamily (mfs) revisited. *The FEBS journal*, 279(11):2022–2035, 2012.
- [170] Peter Aasted Paulsen, Tânia F Custódio, and Bjørn Panyella Pedersen. Crystal structure of the plant symporter stp10 illuminates sugar uptake mechanism in monosaccharide transporter superfamily. *Nature Communications*, 10(1):407, 2019.
- [171] Abdul Aziz Qureshi, Albert Suades, Rei Matsuoka, Joseph Brock, Sarah E McComas, Emmanuel Nji, Laura Orellana, Magnus Claesson, Lucie Delemotte, and David Drew. The molecular basis for sugar import in malaria parasites. *Nature*, 578(7794):321–325, 2020.
- [172] Linfeng Sun, Xin Zeng, Chuangye Yan, Xiuyun Sun, Xinqi Gong, Yu Rao, and Nieng Yan. Crystal structure of a bacterial homologue of glucose transporters glut1–4. *Nature*, 490(7420):361–366, 2012.
- [173] Norimichi Nomura, Grégory Verdon, Hae Joo Kang, Tatsuro Shimamura, Yayoi Nomura, Yo Sonoda, Saba Abdul Hussien, Aziz Abdul Qureshi,

- Mathieu Coincon, Yumi Sato, et al. Structure and mechanism of the mammalian fructose transporter glut5. *Nature*, 526(7573):397–401, 2015.
- [174] Martin CJ Maiden, Elaine O Davis, Stephen A Baldwin, Duncan CM Moore, and Peter JF Henderson. Mammalian and bacterial sugar transport proteins are homologous. *Nature*, 325(6105):641–643, 1987.
- [175] Oded Lewinson, Julia Adler, Gerrit J Poelarends, Piotr Mazurkiewicz, Arnold JM Driessen, and Eitan Bibi. The escherichia coli multidrug transporter mdfa catalyzes both electrogenic and electroneutral transport reactions. *Proceedings of the National Academy of Sciences*, 100(4):1667–1672, 2003.
- [176] Diego Del Alamo, Jens Meiler, and Hassane S Mchaourab. Principles of alternating access in leut-fold transporters: commonalities and divergences. *Journal of Molecular Biology*, 434(19):167746, 2022.
- [177] Åke Vastermark, Simon Wollwage, Michael E Houle, Rita Rio, and Milton H Saier Jr. Expansion of the apc superfamily of secondary carriers. *Proteins: Structure, Function, and Bioinformatics*, 82(10):2797–2811, 2014.
- [178] Palanivel Kandasamy, Gergely Gyimesi, Yoshikatsu Kanai, and Matthias A Hediger. Amino acid transporters revisited: New views in health and disease. *Trends in biochemical sciences*, 43(10):752–789, 2018.
- [179] Åke Vastermark and Milton H Saier Jr. Evolutionary relationship between 5+ 5 and 7+ 7 inverted repeat folds within the amino acid-polyamine-organocation superfamily. *Proteins: Structure, Function, and Bioinformatics*, 82(2):336–346, 2014.
- [180] Atsuko Yamashita, Satinder K Singh, Toshimitsu Kawate, Yan Jin, and Eric Gouaux. Crystal structure of a bacterial homologue of na+/cl- dependent neurotransmitter transporters. *Nature*, 437(7056):215–223, 2005.
- [181] Aravind Penmatsa and Eric Gouaux. How leut shapes our understanding of the mechanisms of sodium-coupled neurotransmitter transporters. *The Journal of physiology*, 592(5):863–869, 2014.
- [182] Noel Edwards, Catriona MH Anderson, Nichola J Conlon, Andrew K Watson, Rebecca J Hall, Timothy R Cheek, T Martin Embley, and David T Thwaites. Resculpting the binding pocket of apc superfamily leut-fold amino acid transporters. *Cellular and Molecular Life Sciences*, 75:921–938, 2018.
- [183] Jeff Abramson and Ernest M Wright. Structure and function of na+-symporters with inverted repeats. *Current opinion in structural biology*, 19(4):425–432, 2009.
- [184] Harini Krishnamurthy, Chayne L Piscitelli, and Eric Gouaux. Unlocking the molecular secrets of sodium-coupled transporters. *Nature*, 459(7245):347–355, 2009.

- [185] Jing Li, Zhiyu Zhao, and Emad Tajkhorshid. Locking two rigid-body bundles in an outward-facing conformation: the ion-coupling mechanism in a leucine transporter. *Scientific reports*, 9(1):19479, 2019.
- [186] Julia Vergalli, Igor V Bodrenko, Muriel Masi, Lucile Moynié, Silvia Acosta-Gutiérrez, James H Naismith, Anne Davin-Regli, Matteo Ceccarelli, Bert van Den Berg, Mathias Winterhalter, et al. Porins and small-molecule translocation across the outer membrane of gram-negative bacteria. *Nature Reviews Microbiology*, 18(3):164–176, 2020.
- [187] Raimund Dutzler, YF Wang, PJ Rizkallah, JP Rosenbusch, and T Schirmer. Crystal structures of various maltooligosaccharides bound to maltoporin reveal a specific sugar translocation pathway. *Structure*, 4(2):127–134, 1996.
- [188] Deepti Chaturvedi and Radhakrishnan Mahalakshmi. Transmembrane β -barrels: Evolution, folding and energetics. *Biochimica et Biophysica Acta (BBA)-Biomembranes*, 1859(12):2467–2482, 2017.
- [189] Daphne Atlas. The voltage-gated calcium channel functions as the molecular switch of synaptic transmission. *Annual review of biochemistry*, 82(1):607–635, 2013.
- [190] Jóhanna Árnadóttir and Martin Chalfie. Eukaryotic mechanosensitive channels. *Annual review of biophysics*, 39(1):111–137, 2010.
- [191] Alex Pautsch and Georg E Schulz. Structure of the outer membrane protein a transmembrane domain. *Nature structural biology*, 5(11):1013–1017, 1998.
- [192] Monika Bayrhuber, Thomas Meins, Michael Habeck, Stefan Becker, Karin Giller, Saskia Villinger, Clemens Vornrhein, Christian Griesinger, Markus Zweckstetter, and Kornelius Zeth. Structure of the human voltage-dependent anion channel. *Proceedings of the National Academy of Sciences*, 105(40):15370–15375, 2008.
- [193] Susan K Buchanan, Barbara S Smith, Lalitha Venkatramani, Di Xia, Lothar Esser, Maya Palnitkar, Ranjan Chakraborty, Dick Van Der Helm, and Johann Deisenhofer. Crystal structure of the outer membrane active transporter fepA from escherichia coli. *Nature structural biology*, 6(1):56–63, 1999.
- [194] Heedeok Hong, Gabor Szabo, and Lukas K Tamm. Electrostatic couplings in ompA ion-channel gating suggest a mechanism for pore opening. *Nature chemical biology*, 2(11):627–635, 2006.
- [195] Etsuko Sugawara and Hiroshi Nikaido. OmpA protein of escherichia coli outer membrane occurs in open and closed channel forms. *Journal of Biological Chemistry*, 269(27):17981–17987, 1994.
- [196] Nathalie Saint, Emmanuelle De, Sylviane Julien, Nicole Orange, and Gérard Molle. Ionophore properties of ompA of escherichia coli. *Biochimica et Biophysica Acta (BBA)-Biomembranes*, 1145(1):119–123, 1993.

- [197] Gene-Wei Li, David Burkhardt, Carol Gross, and Jonathan S Weissman. Quantifying absolute protein synthesis rates reveals principles underlying allocation of cellular resources. *Cell*, 157(3):624–635, 2014.
- [198] Hiroaki Ishida, Alicia Garcia-Herrero, and Hans J Vogel. The periplasmic domain of escherichia coli outer membrane protein a can undergo a localized temperature dependent structural transition. *Biochimica Et Biophysica Acta (BBA)-Biomembranes*, 1838(12):3014–3024, 2014.
- [199] Salete MC Newton, Jennifer S Allen, Zhenghua Cao, Zengbiao Qi, Xun-qing Jiang, Cathy Sprencel, John D Igo, Samuel B Foster, Marvin A Payne, and Phillip E Klebba. Double mutagenesis of a positive charge cluster in the ligand-binding site of the ferric enterobactin receptor, fepa. *Proceedings of the National Academy of Sciences*, 94(9):4560–4565, 1997.
- [200] Li Ma, Wallace Kaserer, Rajasekeran Annamalai, Daniel C Scott, Bo Jin, Xiaoxu Jiang, Qiaobin Xiao, Hossein Maymani, Liliana Moura Massis, Luiz CS Ferreira, et al. Evidence of ball-and-chain transport of ferric enterobactin through fepa. *Journal of Biological Chemistry*, 282(1):397–406, 2007.
- [201] Lorne D Jordan, Yongyao Zhou, Chuck R Smallwood, Yoriko Lill, Ken Ritchie, Wai Tak Yip, Salete M Newton, and Phillip E Klebba. Energy-dependent motion of tonb in the gram-negative bacterial inner membrane. *Proceedings of the National Academy of Sciences*, 110(28):11553–11558, 2013.
- [202] Salete MC Newton, John D Igo, Daniel C Scott, and Phillip E Klebba. Effect of loop deletions on the binding and transport of ferric enterobactin by fepa. *Molecular microbiology*, 32(6):1153–1165, 1999.
- [203] Aritri Majumdar, Vy Trinh, Kyle J Moore, Chuck R Smallwood, Ashish Kumar, Taihao Yang, Daniel C Scott, Noah J Long, Salete M Newton, and Phillip E Klebba. Conformational rearrangements in the n-domain of escherichia coli fepa during ferric enterobactin transport. *Journal of Biological Chemistry*, 295(15):4974–4984, 2020.
- [204] Om P Choudhary, Rachna Ujwal, William Kowallis, Rob Coalson, Jeff Abramson, and Michael Grabe. The electrostatics of vdac: Implications in selectivity and gating. *Biophysical Journal*, 98(3):53a, 2010.
- [205] Marco Colombini. The vdac channel: Molecular basis for selectivity. *Biochimica et Biophysica Acta (BBA)-Molecular Cell Research*, 1863(10):2498–2502, 2016.
- [206] Marco Colombini. Vdac structure, selectivity, and dynamics. *Biochimica et Biophysica Acta (BBA)-Biomembranes*, 1818(6):1457–1465, 2012.
- [207] Yan-Fei Wang, Raimund Dutzler, Pierre J Rizkallah, Jurg P Rosenbusch, and Tilman Schirmer. Channel specificity: structural basis for sugar discrimination and differential flux rates in maltoporin. *Journal of molecular biology*, 272(1):56–63, 1997.

- [208] Tilman Schirmer, Thomas A Keller, Yan-Fei Wang, and Jurg P Rosenbusch. Structural basis for sugar translocation through maltoporin channels at 3.1 Å resolution. *Science*, 267(5197):512–514, 1995.
- [209] Huan-Xiang Zhou and J Andrew McCammon. The gates of ion channels and enzymes. *Trends in biochemical sciences*, 35(3):179–185, 2010.
- [210] Roderick MacKinnon. Potassium channels. *FEBS letters*, 555(1):62–65, 2003.
- [211] Séverine Lobet and Raimund Dutzler. Ion-binding properties of the clc chloride selectivity filter. *The EMBO journal*, 25(1):24–33, 2006.
- [212] Todor Dudev, Boris Musset, Deri Morgan, Vladimir V Cherny, Susan ME Smith, Karine Mazmanian, Thomas E DeCoursey, and Carmay Lim. Selectivity mechanism of the voltage-gated proton channel, hv1. *Scientific reports*, 5(1):10320, 2015.
- [213] Attila Malasics, Dirk Gillespie, Wolfgang Nonner, Douglas Henderson, Bob Eisenberg, and Dezső Boda. Protein structure and ionic selectivity in calcium channels: Selectivity filter size, not shape, matters. *Biochimica et Biophysica Acta (BBA)-Biomembranes*, 1788(12):2471–2480, 2009.
- [214] Raktim N Roy, Kitty Hendriks, Wojciech Kopec, Saeid Abdolvand, Kevin L Weiss, Bert L De Groot, Adam Lange, Han Sun, and Leighton Coates. Structural plasticity of the selectivity filter in a nonselective ion channel. *IUCrJ*, 8(3):421–430, 2021.
- [215] Declan A Doyle, Joao Morais Cabral, Richard A Pfuetzner, Anling Kuo, Jacqueline M Gulbis, Steven L Cohen, Brian T Chait, and Roderick MacKinnon. The structure of the potassium channel: molecular basis of k⁺ conduction and selectivity. *science*, 280(5360):69–77, 1998.
- [216] Roger Milkman. An escherichia coli homologue of eukaryotic potassium channel proteins. *Proceedings of the National Academy of Sciences*, 91(9):3510–3514, 1994.
- [217] Edward SA Humphries and Caroline Dart. Neuronal and cardiovascular potassium channels as therapeutic drug targets: promise and pitfalls. *Journal of biomolecular screening*, 20(9):1055–1073, 2015.
- [218] Qie Kuang, Pasi Purhonen, and Hans Hebert. Structure of potassium channels. *Cellular and molecular life sciences*, 72:3677–3693, 2015.
- [219] Haibo Yu, Sergei Yu Noskov, and Benoît Roux. Two mechanisms of ion selectivity in protein binding sites. *Proceedings of the National Academy of Sciences*, 107(47):20329–20334, 2010.
- [220] Simon Berneche and Benoît Roux. A microscopic view of ion conduction through the k⁺ channel. *Proceedings of the National Academy of Sciences*, 100(15):8644–8648, 2003.
- [221] Huong T Kratochvil, Joshua K Carr, Kimberly Matulef, Alvin W Annen, Hui Li, Michał Maj, Jared Ostmeier, Arnaldo L Serrano, H Raghuraman, Sean D Moran, et al. Instantaneous ion configurations in the k⁺ ion

- channel selectivity filter revealed by 2d ir spectroscopy. *Science*, 353(6303):1040–1044, 2016.
- [222] Simon Berneche and Benoit Roux. Energetics of ion conduction through the k⁺ channel. *Nature*, 414(6859):73–77, 2001.
- [223] Benoît Roux, Simon Bernèche, Bernhard Egwolf, Bogdan Lev, Sergei Y Noskov, Christopher N Rowley, and Haibo Yu. Ion selectivity in channels and transporters. *Journal of general physiology*, 137(5):415–426, 2011.
- [224] Ruo-Xu Gu and Bert L de Groot. Central cavity dehydration as a gating mechanism of potassium channels. *Nature Communications*, 14(1):2178, 2023.
- [225] Joseph D Ho, Ronald Yeh, Andrew Sandstrom, Ilya Chorny, William EC Harries, Rebecca A Robbins, Larry JW Miercke, and Robert M Stroud. Crystal structure of human aquaporin 4 at 1.8 Å and its mechanism of conductance. *Proceedings of the National Academy of Sciences*, 106(18):7437–7442, 2009.
- [226] Abul Kalam Azad, Topu Raihan, Jahed Ahmed, Al Hakim, Tanvir Hossain Emon, and Parveen Afroz Chowdhury. Human aquaporins: functional diversity and potential roles in infectious and non-infectious diseases. *Frontiers in Genetics*, 12:654865, 2021.
- [227] Tamir Gonen and Thomas Walz. The structure of aquaporins. *Quarterly reviews of biophysics*, 39(4):361–396, 2006.
- [228] Elisabeth Kruse, Norbert Uehlein, and Ralf Kaldenhoff. The aquaporins. *Genome biology*, 7:1–6, 2006.
- [229] Abul Kalam Azad, Naoki Yoshikawa, Takahiro Ishikawa, Yoshihiro Sawa, and Hitoshi Shibata. Substitution of a single amino acid residue in the aromatic/arginine selectivity filter alters the transport profiles of tonoplast aquaporin homologs. *Biochimica et Biophysica Acta (BBA)-Biomembranes*, 1818(1):1–11, 2012.
- [230] Eric Beitz, Binghua Wu, Lars M Holm, Joachim E Schultz, and Thomas Zeuthen. Point mutations in the aromatic/arginine region in aquaporin 1 allow passage of urea, glycerol, ammonia, and protons. *Proceedings of the National Academy of Sciences*, 103(2):269–274, 2006.
- [231] Emad Tajkhorshid, Peter Nollert, Morten Ø Jensen, Larry JW Miercke, Joseph O’Connell, Robert M Stroud, and Klaus Schulten. Control of the selectivity of the aquaporin water channel family by global orientational tuning. *Science*, 296(5567):525–530, 2002.
- [232] Stephen D Tyerman, Samantha A McGaughey, Jiaen Qiu, Andrea J Yool, and Caitlin S Byrt. Adaptable and multifunctional ion-conducting aquaporins. *Annual Review of Plant Biology*, 72(1):703–736, 2021.
- [233] Milton H Saier Jr, Vamsee S Reddy, Gabriel Moreno-Hagelsieb, Kevin J Hendargo, Yichi Zhang, Vasu Iddamsetty, Katie Jing Kay Lam, Nuo Tian, Steven Russum, Jianing Wang, et al. The transporter classification database (tcdb): 2021 update. *Nucleic acids research*, 49(D1):D461–D467, 2021.

- [234] Milton H Saier Jr. The bacterial phosphotransferase system: new frontiers 50 years after its discovery. *Journal of molecular microbiology and biotechnology*, 25(2-3):73–78, 2015.
- [235] Jean-Marc Jeckelmann and Bernhard Erni. Carbohydrate transport by group translocation: the bacterial phosphoenolpyruvate: sugar phosphotransferase system. *Bacterial cell walls and membranes*, pages 223–274, 2019.
- [236] Steven C Wang, Pauldeen Davejan, Kevin J Hendargo, Ida Javadi-Razaz, Amy Chou, Daniel C Yee, Faezeh Ghazi, Katie Jing Kay Lam, Adam M Conn, Assael Madrigal, et al. Expansion of the major facilitator superfamily (mfs) to include novel transporters as well as transmembrane-acting enzymes. *Biochimica et Biophysica Acta (BBA)-Biomembranes*, 1862(9):183277, 2020.
- [237] Motonari Tsubaki, Fusako Takeuchi, and Nobuyuki Nakanishi. Cytochrome b561 protein family: expanding roles and versatile transmembrane electron transfer abilities as predicted by a new classification system and protein sequence motif analyses. *Biochimica et Biophysica Acta (BBA)-Proteins and Proteomics*, 1753(2):174–190, 2005.
- [238] Dan Su and Han Asard. Three mammalian cytochromes b561 are ascorbate-dependent ferrireductases. *The FEBS journal*, 273(16):3722–3734, 2006.
- [239] Trevor F Moraes and Reinhart AF Reithmeier. Membrane transport metabolons. *Biochimica et Biophysica Acta (BBA)-Biomembranes*, 1818(11):2687–2706, 2012.
- [240] Hilal Zaid, Ilana Talior-Volodarsky, Costin Antonescu, Zhi Liu, and Amira Klip. Gapdh binds glut4 reciprocally to hexokinase-ii and regulates glucose transport activity. *Biochemical Journal*, 419(2):475–484, 2009.
- [241] James WA Graham, Thomas CR Williams, Megan Morgan, Alisdair R Fernie, R George Ratcliffe, and Lee J Sweetlove. Glycolytic enzymes associate dynamically with mitochondria in response to respiratory demand and support substrate channeling. *The Plant Cell*, 19(11):3723–3738, 2007.
- [242] M Estela Campanella, Haiyan Chu, and Philip S Low. Assembly and regulation of a glycolytic enzyme complex on the human erythrocyte membrane. *Proceedings of the National Academy of Sciences*, 102(7):2402–2407, 2005.
- [243] H BIELKA GDR, N Sharon, and EW Australia. Nomenclature and symbolism for amino acids and peptides. *Pure and Applied Chemistry*, 56:595–624, 1984.
- [244] Nuala A O’Leary, Mathew W Wright, J Rodney Brister, Stacy Ciufu, Diana Haddad, Rich McVeigh, Bhanu Rajput, Barbara Robbertse, Brian Smith-White, Danso Ako-Adjei, et al. Reference sequence (refseq) database at ncbi: current status, taxonomic expansion, and functional annotation. *Nucleic acids research*, 44(D1):D733–D745, 2015.

- [245] Alex Bateman, Maria-Jesus Martin, Sandra Orchard, Michele Magrane, Aduragbemi Adesina, Shadab Ahmad, Emily H Bowler-Barnett, Hema Bye-A-Jee, David Carpentier, Paul Denny, et al. Uniprot: the universal protein knowledgebase in 2025. *Nucleic Acids Research*, 52(D 1):D609–D617, 2024.
- [246] Stacia R Engel, Suzi Aleksander, Robert S Nash, Edith D Wong, Shuai Weng, Stuart R Miyasato, Gavin Sherlock, and J Michael Cherry. Saccharomyces genome database: advances in genome annotation, expanded biochemical pathways, and other key enhancements. *Genetics*, 229(3):iyae185, 2025.
- [247] Leonore Reiser, Erica Bakker, Sabarinath Subramaniam, Xingguo Chen, Swapnil Sawant, Kartik Khosa, Trilok Prithvi, and Tanya Z Berardini. The arabidopsis information resource in 2024. *Genetics*, 227(1):iyae027, 2024.
- [248] Ruth L Seal, Bryony Braschi, Kristian Gray, Tamsin EM Jones, Susan Tweedie, Liora Haim-Vilmovsky, and Elspeth A Bruford. Genenames.org: the hgnc resources in 2023. *Nucleic Acids Research*, 51(D1):D1003–D1009, 2023.
- [249] Richard M Baldarelli, Cynthia L Smith, Martin Ringwald, Joel E Richardson, and Carol J Bult. Mouse genome informatics: an integrated knowledgebase system for the laboratory mouse. *Genetics*, 227(1):iyae031, 2024.
- [250] Yvonne M Bradford, Ceri E Van Slyke, Leyla Ruzicka, Amy Singer, Anne Eagle, David Fashena, Douglas G Howe, Ken Frazer, Ryan Martin, Holly Paddock, et al. Zebrafish information network, the knowledgebase for danio rerio research. *Genetics*, 220(4):iyac016, 2022.
- [251] Tamsin EM Jones, Bethan Yates, Bryony Braschi, Kristian Gray, Susan Tweedie, Ruth L Seal, and Elspeth A Bruford. The vgnc: expanding standardized vertebrate gene nomenclature. *Genome Biology*, 24(1):115, 2023.
- [252] Peter D Karp, Suzanne Paley, Ron Caspi, Anamika Kothari, Markus Krummenacker, Peter E Midford, Lisa R Moore, Pallavi Subhraveti, Socorro Gama-Castro, Victor H Tierrafria, et al. The ecocyc database (2023). *EcoSal Plus*, 11(1):eesp-0002, 2023.
- [253] Rainer Schwacke, Anja Schneider, Eric van der Graaff, Karsten Fischer, Elisabetta Catoni, Marcelo Desimone, Wolf B Frommer, Ulf-Ingo Flügge, and Reinhard Kunze. Aramemnon, a novel database for arabidopsis integral membrane proteins. *Plant physiology*, 131(1):16–26, 2003.
- [254] Donna Maglott, Jim Ostell, Kim D Pruitt, and Tatiana Tatusova. Entrez gene: gene-centered information at ncbi. *Nucleic acids research*, 39(suppl.1):D52–D57, 2010.
- [255] Magali Ruffier, Andreas Kähäri, Monika Komorowska, Stephen Keenan, Matthew Laird, Ian Longden, Glenn Proctor, Steve Searle, Daniel Staines, Kieron Taylor, et al. Ensembl core software resources: storage and programmatic access for dna sequence and genome annotation. *Database*, 2017:bax020, 2017.

- [256] Colman O’Cathail, Alisha Ahamed, Josephine Burgin, Carla Cummins, Rajkumar Devaraj, Khadim Gueye, Dipayan Gupta, Vikas Gupta, Muhammad Haseeb, Maira Ihsan, et al. The european nucleotide archive in 2024. *Nucleic Acids Research*, 53(D1):D49–D55, 2025.
- [257] Eric W Sayers, Mark Cavanaugh, Karen Clark, Kim D Pruitt, Conrad L Schoch, Stephen T Sherry, and Ilene Karsch-Mizrachi. Genbank. *Nucleic acids research*, 50(D1):D161–D164, 2022.
- [258] Jun Mashima, Yuichi Kodama, Takatomo Fujisawa, Toshiaki Katayama, Yoshihiro Okuda, Eli Kaminuma, Osamu Ogasawara, Kousaku Okubo, Yasukazu Nakamura, and Toshihisa Takagi. Dna data bank of japan. *Nucleic Acids Research*, page gkw1001, 2016.
- [259] Guy Cochrane, Ilene Karsch-Mizrachi, Toshihisa Takagi, and International Nucleotide Sequence Database Collaboration. The international nucleotide sequence database collaboration. *Nucleic acids research*, 44(D1):D48–D50, 2016.
- [260] Daniel R Zerbino, Premanand Achuthan, Wasiu Akanni, M Ridwan Amode, Daniel Barrell, Jyothish Bhai, Konstantinos Billis, Carla Cummins, Astrid Gall, Carlos García Girón, et al. Ensembl 2018. *Nucleic acids research*, 46(D1):D754–D761, 2017.
- [261] Gerardo Perez, Galt P Barber, Anna Benet-Pages, Jonathan Casper, Hiram Clawson, Mark Diekhans, Clay Fischer, Jairo Navarro Gonzalez, Angie S Hinrichs, Christopher M Lee, et al. The ucsc genome browser database: 2025 update. *Nucleic Acids Research*, 53(D1):D1243–D1249, 2025.
- [262] Michael Y Galperin, Roberto Vera Alvarez, Svetlana Karamycheva, Kira S Makarova, Yuri I Wolf, David Landsman, and Eugene V Koonin. Cog database update 2024. *Nucleic Acids Research*, 53(D1):D356–D363, 2025.
- [263] Sanjida H Rangwala, Anatoliy Kuznetsov, Victor Ananiev, Andrea Asztalos, Evgeny Borodin, Vladislav Evgeniev, Victor Joukov, Vadim Lotov, Ravinder Pannu, Dmitry Rudnev, et al. Accessing ncbi data using the ncbi sequence viewer and genome data viewer (gdv). *Genome research*, 31(1):159–169, 2021.
- [264] Helen M Berman, John Westbrook, Zukang Feng, Gary Gilliland, Talapady N Bhat, Helge Weissig, Ilya N Shindyalov, and Philip E Bourne. The protein data bank. *Nucleic acids research*, 28(1):235–242, 2000.
- [265] Where do the UniProtKB protein sequences come from? Uniprot Help, https://www.uniprot.org/help/sequence_origin, . Accessed: 28.03.2025.
- [266] What is the canonical sequence? Are all isoforms described in one entry? Uniprot Help, https://www.uniprot.org/help/canonical_and_isoforms, . Accessed: 28.03.2025.
- [267] Baris E Suzek, Yuqi Wang, Hongzhan Huang, Peter B McGarvey, Cathy H Wu, and UniProt Consortium. Uniref clusters: a comprehensive and scalable alternative for improving sequence similarity searches. *Bioinformatics*, 31(6):926–932, 2015.

- [268] What is a UniRef seed sequence compared to a representative sequence? Uniprot Help, https://www.uniprot.org/help/uniref_seed, . Accessed: 28.03.2025.
- [269] Rasko Leinonen, Federico Garcia Diez, David Binns, Wolfgang Fleischmann, Rodrigo Lopez, and Rolf Apweiler. Uniprot archive. *Bioinformatics*, 20(17):3236–3237, 2004.
- [270] Helen Berman, Kim Henrick, and Haruki Nakamura. Announcing the worldwide protein data bank. *Nature structural & molecular biology*, 10(12):980–980, 2003.
- [271] Stephen K Burley, Helen M Berman, Jose M Duarte, Zukang Feng, Justin W Flatt, Brian P Hudson, Robert Lowe, Ezra Peisach, Dennis W Piehl, Yana Rose, et al. Protein data bank: a comprehensive review of 3d structure holdings and worldwide utilization by researchers, educators, and students. *Biomolecules*, 12(10):1425, 2022.
- [272] Stephen K Burley, Charmi Bhikadiya, Chunxiao Bi, Sebastian Bittrich, Henry Chao, Li Chen, Paul A Craig, Gregg V Crichlow, Kenneth Dalenberg, Jose M Duarte, et al. Rcsb protein data bank (rcsb.org): delivery of experimentally-determined pdb structures alongside one million computed structure models of proteins from artificial intelligence/machine learning. *Nucleic acids research*, 51(D1):D488–D508, 2023.
- [273] Gerardo Tauriello, Andrew M Waterhouse, Juergen Haas, Dario Behringer, Stefan Bienert, Thomas Garello, and Torsten Schwede. Modelarchive: a deposition database for computational macromolecular structural models. *Journal of Molecular Biology*, page 168996, 2025.
- [274] Mihaly Varadi, Stephen Anyango, Mandar Deshpande, Sreenath Nair, Cindy Natassia, Galabina Yordanova, David Yuan, Oana Stroe, Gemma Wood, Agata Laydon, Augustin Židek, Tim Green, Kathryn Tunyasuvunakool, Stig Petersen, John Jumper, Ellen Clancy, Richard Green, Ankur Vora, Mira Lutfi, Michael Figurnov, Andrew Cowie, Nicole Hobbs, Pushmeet Kohli, Gerard Kleywegt, Ewan Birney, Demis Hassabis, and Sameer Velankar. Alphafold protein structure database: Massively expanding the structural coverage of protein-sequence space with high-accuracy models. *Nucleic Acids Res.*, 50(D1):D439–D444, 2022.
- [275] Mihaly Varadi, Damian Bertoni, Paulyna Magana, Urmila Paramval, Ivanna Pidruchna, Malarvizhi Radhakrishnan, Maxim Tsenkov, Sreenath Nair, Milot Mirdita, Jingsi Yeo, et al. Alphafold protein structure database in 2024: providing structure coverage for over 214 million protein sequences. *Nucleic acids research*, 52(D1):D368–D375, 2024.
- [276] Robert M Hanson, Jaime Prilusky, Zhou Renjian, Takanori Nakane, and Joel L Sussman. Jsmol and the next-generation web-based representation of 3d molecular structure as applied to proteopedia. *Israel Journal of Chemistry*, 53(3-4):207–216, 2013.
- [277] Thomas D Newport, Mark S P Sansom, and Phillip J Stansfeld. The memprotmd database: a resource for membrane-embedded protein structures

- and their lipid interactions. *Nucleic acids research*, 47(D1):D390–D397, 2019.
- [278] Daniel Kozma, Istvan Simon, and Gabor E Tusnady. Pdbtm: Protein data bank of transmembrane proteins after 8 years. *Nucleic acids research*, 41(D1):D524–D529, 2012.
- [279] Retrieved from QuickGO Gene Ontology Browser. QuickGO, <https://www.ebi.ac.uk/QuickGO/>. Accessed: 06.04.2025, GO version 2025-04-06, Annotation set created on 2025-03-06 17:15.
- [280] Barry Smith, Michael Ashburner, Cornelius Rosse, Jonathan Bard, William Bug, Werner Ceusters, Louis J Goldberg, Karen Eilbeck, Amelia Ireland, Christopher J Mungall, et al. The obo foundry: coordinated evolution of ontologies to support biomedical data integration. *Nature biotechnology*, 25(11):1251–1255, 2007.
- [281] OWL 2 Web Ontology Language Document Overview (Second Edition). W3C website, <https://www.w3.org/TR/owl2-overview/>. Accessed: 06.04.2025.
- [282] Birte Glimm, Ian Horrocks, Boris Motik, Giorgos Stoilos, and Zhe Wang. Hermit: an owl 2 reasoner. *Journal of automated reasoning*, 53:245–269, 2014.
- [283] Yevgeny Kazakov, Markus Krötzsch, and František Simančík. Elk: a reasoner for owl el ontologies. *System Description*, pages 49–64, 2012.
- [284] Rebecca C Jackson, James P Balhoff, Eric Douglass, Nomi L Harris, Christopher J Mungall, and James A Overton. Robot: a tool for automating ontology workflows. *BMC bioinformatics*, 20:1–10, 2019.
- [285] Richard G Côté, Philip Jones, Rolf Apweiler, and Henning Hermjakob. The ontology lookup service, a lightweight cross-platform tool for controlled vocabulary queries. *BMC bioinformatics*, 7:1–7, 2006.
- [286] Mark A Musen. The protégé project: a look back and a look forward. *AI matters*, 1(4):4–12, 2015.
- [287] Matthew Horridge and Sean Bechhofer. The owl api: A java api for owl ontologies. *Semantic web*, 2(1):11–21, 2011.
- [288] Rachael P Huntley, Tony Sawford, Prudence Mutowo-Meullenet, Aleksandra Shypitsyna, Carlos Bonilla, Maria J Martin, and Claire O’Donovan. The goa database: gene ontology annotation updates for 2015. *Nucleic acids research*, 43(D1):D1057–D1063, 2015.
- [289] David Binns, Emily Dimmer, Rachael Huntley, Daniel Barrell, Claire O’donovan, and Rolf Apweiler. Quickgo: a web-based tool for gene ontology searching. *Bioinformatics*, 25(22):3045–3046, 2009.
- [290] Seth Carbon, Amelia Ireland, Christopher J Mungall, ShengQiang Shu, Brad Marshall, Suzanna Lewis, AmiGO Hub, and Web Presence Working Group. Amigo: online access to ontology and annotation data. *Bioinformatics*, 25(2):288–289, 2009.

- [291] Rolf Apweiler, Amos Bairoch, Cathy H Wu, Winona C Barker, Brigitte Boeckmann, Serenella Ferro, Elisabeth Gasteiger, Hongzhan Huang, Rodrigo Lopez, Michele Magrane, et al. Uniprot: the universal protein knowledgebase. *Nucleic acids research*, 32(suppl_1):D115–D119, 2004.
- [292] Kirill Degtyarenko, Paula De Matos, Marcus Ennis, Janna Hastings, Martin Zbinden, Alan McNaught, Rafael Alcántara, Michael Darsow, Mickaël Guedj, and Michael Ashburner. Chebi: a database and ontology for chemical entities of biological interest. *Nucleic Acids Res.*, 36(suppl_1):D344–D350, 2007.
- [293] Retrieved from ChEBI browser. ChEBI, <https://www.ebi.ac.uk/chebi/>. Accessed: 06.04.2025, ChEBI version 241.
- [294] Matthias Blum, Antonina Andreeva, Laise Cavalcanti Florentino, Sara Rocio Chuguransky, Tiago Grego, Emma Hobbs, Beatriz Lazaro Pinto, Ailsa Orr, Typhaine Paysan-Lafosse, Irina Ponamareva, et al. Interpro: the protein sequence classification resource in 2025? mode longmeta?;. *Nucleic Acids Research*, 53(D1):D444–D456, 2025.
- [295] Philip Jones, David Binns, Hsin-Yu Chang, Matthew Fraser, Weizhong Li, Craig McAnulla, Hamish McWilliam, John Maslen, Alex Mitchell, Gift Nuka, et al. Interproscan 5: genome-scale protein function classification. *Bioinformatics*, 30(9):1236–1240, 2014.
- [296] Milton H Saier Jr, Vamsee S Reddy, Brian V Tsu, Muhammad Saad Ahmed, Chun Li, and Gabriel Moreno-Hagelsieb. The transporter classification database (tcdb): recent advances. *Nucleic acids research*, 44(D1):D372–D379, 2015.
- [297] Liam DH Elbourne, Sasha G Tetu, Karl A Hassan, and Ian T Paulsen. Transportdb 2.0: a database for exploring membrane transporters in sequenced genomes from all domains of life. *Nucleic acids research*, 45(D1):D320–D324, 2016.
- [298] Liam DH Elbourne, Brendan Wilson-Mortier, Qinghu Ren, Karl A Hassan, Sasha G Tetu, and Ian T Paulsen. Transaap: an automated annotation pipeline for membrane transporter prediction in bacterial genomes. *Microbial genomics*, 9(1):000927, 2023.
- [299] Stephen F Altschul, Thomas L Madden, Alejandro A Schäffer, Jinghui Zhang, Zheng Zhang, Webb Miller, and David J Lipman. Gapped blast and psi-blast: a new generation of protein database search programs. *Nucleic acids research*, 25(17):3389–3402, 1997.
- [300] Simon C Potter, Aurélien Luciani, Sean R Eddy, Youngmi Park, Rodrigo Lopez, and Robert D Finn. Hmmer web server: 2018 update. *Nucleic acids research*, 46(W1):W200–W204, 2018.
- [301] Susan Tweedie, Bryony Braschi, Kristian Gray, Tamsin EM Jones, Ruth L Seal, Bethan Yates, and Elspeth A Bruford. Genenames.org: the hgnc and vgnc resources in 2021. *Nucleic Acids Res.*, 49(D1):D939–D946, 2021.

- [302] HGNC gene group 752 Solute carrier families (SLC). HGNC website, <https://www.genenames.org/data/genegroup/#!/group/752>, . Accessed: 08.04.2025.
- [303] HGNC gene group 2155 Solute carrier family 2 (SLC2), also known as Facilitative GLUT transporter family. HGNC website, <https://www.genenames.org/data/genegroup/#!/group/2155>, . Accessed: 08.04.2025.
- [304] HGNC gene group 417 ATP binding cassette transporters (ABC). HGNC website, <https://www.genenames.org/data/genegroup/#!/group/417>, . Accessed: 08.04.2025.
- [305] HGNC gene group 412 ATPases. HGNC website, <https://www.genenames.org/data/genegroup/#!/group/412>, . Accessed: 08.04.2025.
- [306] HGNC gene group 177 Ion channels. HGNC website, <https://www.genenames.org/data/genegroup/#!/group/177>, . Accessed: 08.04.2025.
- [307] Simon D Harding, Jane F Armstrong, Elena Faccenda, Christopher Southan, Stephen PH Alexander, Anthony P Davenport, Michael Spedding, and Jamie A Davies. The iuphar/bps guide to pharmacology in 2024. *Nucleic acids research*, 52(D1):D1438–D1449, 2024.
- [308] Craig Knox, Mike Wilson, Christen M Klinger, Mark Franklin, Eponine Oler, Alex Wilson, Allison Pon, Jordan Cox, Na Eun Chin, Seth A Strawbridge, et al. Drugbank 6.0: the drugbank knowledgebase for 2024. *Nucleic acids research*, 52(D1):D1265–D1275, 2024.
- [309] MO Dayhoff, RM Schwartz, and BC Orcutt. 22 a model of evolutionary change in proteins. *Atlas of protein sequence and structure*, pages 345–352, 1978.
- [310] Sean R Eddy. Where did the blosum62 alignment score matrix come from? *Nature biotechnology*, 22(8):1035–1036, 2004.
- [311] Saul B Needleman and Christian D Wunsch. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *J. Mol. Biol.*, 48(3):443–453, 1970.
- [312] Temple F Smith, Michael S Waterman, et al. Identification of common molecular subsequences. *Journal of molecular biology*, 147(1):195–197, 1981.
- [313] Daniel S. Hirschberg. A linear space algorithm for computing maximal common subsequences. *Communications of the ACM*, 18(6):341–343, 1975.
- [314] NCBI non-redundant protein database (nr). All non-redundant GenBank CDS translations + PDB + SwissProt + PIR + PRF excluding environmental samples from WGS projects. NCBI Protein BLAST website, <https://blast.ncbi.nlm.nih.gov/Blast.cgi>. Accessed: 10.04.2025.
- [315] Samuel Karlin and Stephen F Altschul. Methods for assessing the statistical significance of molecular sequence features by using general scoring schemes. *Proceedings of the National Academy of Sciences*, 87(6):2264–2268, 1990.

- [316] Christiam Camacho, George Coulouris, Vahram Avagyan, Ning Ma, Jason Papadopoulos, Kevin Bealer, and Thomas L Madden. Blast+: architecture and applications. *BMC Bioinf.*, 10(1):1–9, 2009.
- [317] Weizhong Li and Adam Godzik. Cd-hit: a fast program for clustering and comparing large sets of protein or nucleotide sequences. *Bioinformatics*, 22(13):1658–1659, 2006.
- [318] Jaime Huerta-Cepas, Damian Szklarczyk, Davide Heller, Ana Hernández-Plaza, Sofia K Forslund, Helen Cook, Daniel R Mende, Ivica Letunic, Thomas Rattei, Lars J Jensen, et al. eggno5.0: a hierarchical, functionally and phylogenetically annotated orthology resource based on 5090 organisms and 2502 viruses. *Nucleic acids research*, 47(D1):D309–D314, 2019.
- [319] UniRef Help Page. Uniprot Help, <https://www.uniprot.org/help/uniref>, . Accessed: 28.03.2025.
- [320] What is a UniRef cluster? Uniprot Help, https://www.uniprot.org/help/uniref_cluster, . Accessed: 28.03.2025.
- [321] Jerzy Neyman and Egon Sharpe Pearson. Ix. on the problem of the most efficient tests of statistical hypotheses. *Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character*, 231(694-706):289–337, 1933.
- [322] Ronald Aylmer Fisher. Statistical methods for research workers. In *Breakthroughs in statistics: Methodology and distribution*, pages 66–70. Springer, 1970.
- [323] Peter Bruce, Andrew Bruce, and Peter Gedeck. *Practical statistics for data scientists: 50+ essential concepts using R and Python*. O’Reilly Media, 2020.
- [324] Dennis D Wackerly. *Mathematical statistics with applications*. Thomson Brooks/Cole, 2008.
- [325] Aurélien Géron. *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: Concepts, tools, and techniques to build intelligent systems*. ” O’Reilly Media, Inc.”, 2022.
- [326] Carlo Bonferroni. Teoria statistica delle classi e calcolo delle probabilita. *Pubblicazioni del R istituto superiore di scienze economiche e commerciali di firenze*, 8:3–62, 1936.
- [327] S. S. SHAPIRO and M. B. WILK. An analysis of variance test for normality (complete samples). *Biometrika*, 52(3-4):591–611, dec 1965. doi: 10.1093/biomet/52.3-4.591. URL <https://doi.org/10.1093/biomet/52.3-4.591>.
- [328] J Patrick Royston. An extension of shapiro and wilk’s w test for normality to large samples. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 31(2):115–124, 1982.

- [329] Patrick Royston. Remark as r94: A remark on algorithm as 181: The w-test for normality. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 44(4):547–551, 1995.
- [330] Bee Wah Yap and Chiaw Hock Sim. Comparisons of various types of normality tests. *Journal of Statistical Computation and Simulation*, 81(12):2141–2155, 2011.
- [331] Nornadiah Mohd Razali, Yap Bee Wah, et al. Power comparisons of shapiro-wilk, kolmogorov-smirnov, lilliefors and anderson-darling tests. *Journal of statistical modeling and analytics*, 2(1):21–33, 2011.
- [332] Student. The probable error of a mean. *Biometrika*, pages 1–25, 1908.
- [333] Bernard L Welch. The generalization of ‘student’s’ problem when several different population variances are involved. *Biometrika*, 34(1-2):28–35, 1947.
- [334] David S Moore and Stephane Kirkland. *The basic practice of statistics*, volume 2. WH Freeman New York, 2007.
- [335] Frank Wilcoxon. Individual comparisons by ranking methods. In *Break-throughs in statistics: Methodology and distribution*, pages 196–202. Springer, 1945,1992.
- [336] Pauli Virtanen, Ralf Gommers, Travis E Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, et al. Scipy 1.0: fundamental algorithms for scientific computing in python. *Nature methods*, 17(3):261–272, 2020.
- [337] SciPy documentation for Mann-Whitney U test. SciPy version 1.15.2 reference, <https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.mannwhitneyu.html>. Accessed: 18.04.2025.
- [338] Patrick Schober, Christa Boer, and Lothar A Schwarte. Correlation coefficients: appropriate use and interpretation. *Anesthesia & analgesia*, 126(5):1763–1768, 2018.
- [339] Charles Spearman. The proof and measurement of association between two things. 1961.
- [340] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20:273–297, 1995.
- [341] Bernhard E Boser, Isabelle M Guyon, and Vladimir N Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the fifth annual workshop on Computational learning theory*, pages 144–152, 1992.
- [342] Chih-Chung Chang and Chih-Jen Lin. Libsvm: a library for support vector machines. *ACM transactions on intelligent systems and technology (TIST)*, 2(3):27, 2011.
- [343] Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. Liblinear: A library for large linear classification. *the Journal of machine Learning research*, 9:1871–1874, 2008.

- [344] Tianbao Yang, Yu-Feng Li, Mehrdad Mahdavi, Rong Jin, and Zhi-Hua Zhou. Nyström method vs random fourier features: A theoretical and empirical comparison. *Advances in neural information processing systems*, 25, 2012.
- [345] Christopher Williams and Matthias Seeger. Using the nyström method to speed up kernel machines. *Advances in neural information processing systems*, 13, 2000.
- [346] Ali Rahimi and Benjamin Recht. Random features for large-scale kernel machines. *Advances in neural information processing systems*, 20, 2007.
- [347] John Platt et al. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. *Advances in large margin classifiers*, 10(3):61–74, 1999.
- [348] Ting-Fan Wu, Chih-Jen Lin, and Ruby C Weng. Probability estimates for multi-class classification by pairwise coupling. *Journal of Machine Learning Research*, 5(Aug):975–1005, 2004.
- [349] Andreas Denger and Volkhard Helms. Optimized data set and feature construction for substrate prediction of membrane transporters. *Journal of Chemical Information and Modeling*, 62(23):6242–6257, 2022.
- [350] Sarah Hunter, Rolf Apweiler, Teresa K Attwood, Amos Bairoch, Alex Bateman, David Binns, Peer Bork, Ujjwal Das, Louise Daugherty, Lorraine Duquenne, et al. Interpro: the integrative protein signature database. *Nucleic acids research*, 37(suppl_1):D211–D215, 2009.
- [351] Vladimir Gligorijević, P Douglas Renfrew, Tomasz Kosciolk, Julia Koehler Leman, Daniel Berenberg, Tommi Vatanen, Chris Chandler, Bryn C Taylor, Ian M Fisk, Hera Vlamakis, et al. Structure-based protein function prediction using graph convolutional networks. *Nature communications*, 12(1):3168, 2021.
- [352] Guillaume Lemaître, Fernando Nogueira, and Christos K Aridas. Imbalanced-learn: A python toolbox to tackle the curse of imbalanced datasets in machine learning. *Journal of machine learning research*, 18 (17):1–5, 2017.
- [353] Joe H Ward Jr. Hierarchical grouping to optimize an objective function. *Journal of the American statistical association*, 58(301):236–244, 1963.
- [354] Karl Pearson. Liii. on lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin philosophical magazine and journal of science*, 2(11):559–572, 1901.
- [355] Ji Zhu, Saharon Rosset, Robert Tibshirani, and Trevor Hastie. 1-norm support vector machines. *Advances in neural information processing systems*, 16, 2003.
- [356] Karl Pearson. X. on the criterion that a given system of deviations from the probable in the case of a correlated system of variables is such that it can be reasonably supposed to have arisen from random sampling. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 50(302):157–175, 1900.

- [357] David MW Powers. Evaluation: from precision, recall and f-measure to roc, informedness, markedness and correlation. *arXiv preprint arXiv:2010.16061*, 2020.
- [358] Kay Henning Brodersen, Cheng Soon Ong, Klaas Enno Stephan, and Joachim M Buhmann. The balanced accuracy and its posterior distribution. In *2010 20th international conference on pattern recognition*, pages 3121–3124. IEEE, 2010.
- [359] Brian W Matthews. Comparison of the predicted and observed secondary structure of t4 phage lysozyme. *Biochimica et Biophysica Acta (BBA)-Protein Structure*, 405(2):442–451, 1975.
- [360] UniProt Consortium. Uniprot: a worldwide hub of protein knowledge. *Nucleic Acids Res.*, 47(D1):D506–D515, 2019.
- [361] Rosalin Bonetta and Gianluca Valentino. Machine learning techniques for protein function prediction. *Proteins: Struct., Funct., Bioinf.*, 88(3): 397–413, 2020.
- [362] M Michael Gromiha and Yukimitsu Yabuki. Functional discrimination of membrane proteins using learning techniques. *BMC bioinformatics*, 9(1): 135, 2008.
- [363] M Michael Gromiha and Yu-Yen Ou. Bioinformatics approaches for functional annotation of membrane proteins. *Briefings in bioinformatics*, 15 (2):155–168, 2013.
- [364] Aina W Ravna, Georg Sager, Svein G Dahl, and Ingebrigt Sylte. Membrane transporters: structure, function and targets for drug design. In *Transporters as targets for drugs*, pages 15–51. Springer, 2008.
- [365] Russell F Doolittle. Similar amino acid sequences: chance or common ancestry? *Science*, 214(4517):149–159, 1981.
- [366] NS Schaadt and V Helms. Functional classification of membrane transporters and channels based on filtered tm/non-tm amino acid composition. *Biopolymers*, 97(7):558–567, 2012.
- [367] Munira Alballa, Faizah Aplop, and Gregory Butler. Trancep: Predicting the substrate class of transmembrane transport proteins using compositional, evolutionary, and positional information. *PloS one*, 15(1):e0227683, 2020.
- [368] Hiroshi Nakashima and Ken Nishikawa. Discrimination of intracellular and extracellular proteins using amino acid composition and residue-pair frequencies. *Journal of molecular biology*, 238(1):54–61, 1994.
- [369] Gary D Stormo, Thomas D Schneider, Larry Gold, and Andrzej Ehrenfeucht. Use of the ‘perceptron’ algorithm to distinguish translational initiation sites in e. coli. *Nucleic acids research*, 10(9):2997–3011, 1982.
- [370] Shu-An Chen, Yu-Yen Ou, Tzong-Yi Lee, and M Michael Gromiha. Prediction of transporter targets using efficient rbf networks with pssm profiles and biochemical properties. *Bioinformatics*, 27(15):2062–2067, 2011.

- [371] Guillaume Pilot, Harald Stransky, Dean F Bushey, Reéjane Pratelli, Uwe Ludewig, Vincent PM Wingate, and Wolf B Frommer. Overexpression of glutamine dumper1 leads to hypersecretion of glutamine from hydathodes of arabidopsis leaves. *The Plant Cell*, 16(7):1827–1840, 2004.
- [372] Réjane Pratelli and Guillaume Pilot. The plant-specific vimag domain of glutamine dumper1 is necessary for the function of the protein in arabidopsis. *FEBS Lett.*, 580(30):6961–6966, 2006.
- [373] Rejane Pratelli, Lars M Voll, Robin J Horst, Wolf B Frommer, and Guillaume Pilot. Stimulation of nonselective amino acid export by glutamine dumper proteins. *Plant Physiol.*, 152(2):762–773, 2010.
- [374] Frederic Basyn, Benoit Spies, Olivier Bouffieux, Annick Thomas, and Robert Brasseur. Insertion of x-ray structures of proteins in membranes. *J. Mol. Graphics Modell.*, 22(1):11–21, 2003.
- [375] Vu Ha Tran, Ahmad Barghash, and Volkhard Helms. Annotating the function of protein-coding genes based on gene ontology terms of neighboring co-expressed genes. *Journal of Proteomics and Bioinformatics*, 11: 068–074, 2018.
- [376] Andreas Denger and Volkhard Helms. Identifying optimal substrate classes of membrane transporters. *PloS one*, 19(12):e0315330, 2024.
- [377] Membrane transporters in drug development. *Nat. Rev. Drug Discovery*, 9(3):215–236, 2010.
- [378] Ahmed Elnaggar, Michael Heinzinger, Christian Dallago, Ghalia Rehawi, Yu Wang, Llion Jones, Tom Gibbs, Tamas Feher, Christoph Angerer, Martin Steinegger, et al. Prottrans: Toward understanding the language of life through self-supervised learning. *IEEE Trans. Pattern Anal. Mach. Intell.*, 44(10):7112–7127, 2021.
- [379] George Diallinas. Understanding transporter specificity and the discrete appearance of channel-like gating domains in transporters. *Frontiers in pharmacology*, 5:105861, 2014.
- [380] Nitin Gupta, Shashank Mujumdar, Hima Patel, Satoshi Masuda, Naveen Panwar, Sambaran Bandyopadhyay, Sameep Mehta, Shanmukha Guttula, Shazia Afzal, Ruhi Sharma Mittal, et al. Data quality for machine learning tasks. In *Proceedings of the 27th ACM SIGKDD conference on knowledge discovery & data mining*, pages 4040–4041, 2021.
- [381] Stuart G Jantzen, Ben JG Sutherland, David R Minkley, and Ben F Koop. Go trimming: Systematically reducing redundancy in large gene ontology datasets. *BMC research notes*, 4:1–9, 2011.
- [382] Sebastian Bauer, Steffen Grossmann, Martin Vingron, and Peter N Robinson. Ontologizer 2.0—a multifunctional tool for go term enrichment analysis and data exploration. *Bioinformatics*, 24(14):1650–1651, 2008.
- [383] Guangchuang Yu, Fei Li, Yide Qin, Xiaochen Bo, Yibo Wu, and Shengqi Wang. Gosemsim: an r package for measuring semantic similarity among go terms and gene products. *Bioinformatics*, 26(7):976–978, 2010.

- [384] James Z Wang, Zhidian Du, Rapeeporn Payattakool, Philip S Yu, and Chin-Fu Chen. A new method to measure the semantic similarity of go terms. *Bioinformatics*, 23(10):1274–1281, 2007.
- [385] Pietro H Guzzi, Marco Mina, Concettina Guerra, and Mario Cannataro. Semantic similarity analysis of protein data: assessment with biological features and issues. *Briefings in bioinformatics*, 13(5):569–585, 2012.
- [386] Chenguang Zhao and Zheng Wang. Gogo: An improved algorithm to measure the semantic similarity between gene ontology terms. *Scientific reports*, 8(1):15107, 2018.
- [387] David Weininger. Smiles, a chemical language and information system. 1. introduction to methodology and encoding rules. *J. Chem. Inf. Comput. Sci.*, 28(1):31–36, 1988.
- [388] Greg Landrum. Rdkit: Open-source cheminformatics. <https://www.rdkit.org>. 2023. URL <https://doi.org/10.5281/zenodo.8053810>.
- [389] Harry L Morgan. The generation of a unique machine description for chemical structures-a technique developed at chemical abstracts service. *Journal of chemical documentation*, 5(2):107–113, 1965.
- [390] Raymond E Carhart, Dennis H Smith, and R Venkataraghavan. Atom pairs as molecular features in structure-activity studies: definition and applications. *J. Chem. Inf. Comput. Sci.*, 25(2):64–73, 1985.
- [391] Ramaswamy Nilakantan, Norman Bauman, J Scott Dixon, and R Venkataraghavan. Topological torsion: a new molecular descriptor for sar applications. comparison with other descriptors. *J. Chem. Inf. Comput. Sci.*, 27(2):82–85, 1987.
- [392] MDL MACCS-II. Information systems/symyx. *Santa Clara, CA*, 1984.
- [393] Taffee T Tanimoto. Elementary mathematical theory of classification and prediction. 1958.
- [394] Baris E Suzek, Hongzhan Huang, Peter McGarvey, Raja Mazumder, and Cathy H Wu. Uniref: comprehensive and non-redundant uniprot reference clusters. *Bioinformatics*, 23(10):1282–1288, 2007.
- [395] Robert Schmidt, Emanuel SR Ehmki, Farina Ohm, Hans-Christian Ehrlich, Andriy Mashychev, and Matthias Rarey. Comparing molecular patterns using the example of smarts: theory and algorithms. *Journal of Chemical Information and Modeling*, 59(6):2560–2571, 2019.
- [396] Andreas Denger and Volkhard Helms. Application of protein structure encodings and sequence embeddings for transporter substrate prediction. *Molecules*, 30(15):3226, 2025.
- [397] Evandro Ferrada and Giulio Superti-Furga. A structure and evolutionary-based classification of solute carriers. *IScience*, 25(10), 2022.
- [398] Li-Qing Chen, Bi-Huei Hou, Sylvie Lalonde, Hitomi Takanaga, Mara L Hartung, Xiao-Qing Qu, Woei-Jiun Guo, Jung-Gun Kim, William Un-

- derwood, Bhavna Chaudhuri, et al. Sugar transporters for intercellular exchange and nutrition of pathogens. *Nature*, 468(7323):527–532, 2010.
- [399] Jialei Ji, Limei Yang, Zhiyuan Fang, Yangyong Zhang, Mu Zhuang, Honghao Lv, and Yong Wang. Plant sweet family of sugar transporters: structure, evolution and biological functions. *Biomolecules*, 12(2):205, 2022.
- [400] Alexander Kroll, Nico Niebuhr, Gregory Butler, and Martin J Lercher. Spot: A machine learning model that predicts specific substrates for transport proteins. *PLoS biology*, 22(9):e3002807, 2024.
- [401] Muhammad Shahid Malik, Van The Le, and Yu-Yen Ou. Na_mcn: Classification of sodium transporters in membrane proteins by integrating multi-window deep learning and prottrans for their therapeutic potential. *Journal of Proteome Research*, 24(5):2324–2335, 2025.
- [402] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.
- [403] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 2002.
- [404] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE transactions on neural networks*, 20(1):61–80, 2008.
- [405] Diederik P Kingma, Max Welling, et al. Auto-encoding variational bayes, 2013.
- [406] Alexander Rives, Joshua Meier, Tom Sercu, Siddharth Goyal, Zeming Lin, Jason Liu, Demi Guo, Myle Ott, C Lawrence Zitnick, Jerry Ma, et al. Biological structure and function emerge from scaling unsupervised learning to 250 million protein sequences. *Proceedings of the National Academy of Sciences*, 118(15):e2016239118, 2021.
- [407] Ali Madani, Ben Krause, Eric R Greene, Subu Subramanian, Benjamin P Mohr, James M Holton, Jose Luis Olmos Jr, Caiming Xiong, Zachary Z Sun, Richard Socher, et al. Large language models generate functional protein sequences across diverse families. *Nature biotechnology*, 41(8):1099–1106, 2023.
- [408] Noelia Ferruz, Steffen Schmidt, and Birte Höcker. Protgpt2 is a deep unsupervised language model for protein design. *Nature communications*, 13(1):4348, 2022.
- [409] Robert Verkuil, Ori Kabeli, Yilun Du, Basile IM Wicky, Lukas F Milles, Justas Dauparas, David Baker, Sergey Ovchinnikov, Tom Sercu, and Alexander Rives. Language models generalize beyond natural proteins. *BioRxiv*, pages 2022–12, 2022.
- [410] Zeming Lin, Halil Akin, Roshan Rao, Brian Hie, Zhongkai Zhu, Wenting Lu, Nikita Smetanin, Robert Verkuil, Ori Kabeli, Yaniv Shmueli, et al.

- Evolutionary-scale prediction of atomic-level protein structure with a language model. *Science*, 379(6637):1123–1130, 2023.
- [411] Minkyung Baek, Frank DiMaio, Ivan Anishchenko, Justas Dauparas, Sergey Ovchinnikov, Gyu Rie Lee, Jue Wang, Qian Cong, Lisa N Kinch, R Dustin Schaeffer, et al. Accurate prediction of protein structures and interactions using a three-track neural network. *Science*, 373(6557):871–876, 2021.
- [412] Michael Heinzinger, Maria Littmann, Ian Sillitoe, Nicola Bordin, Christine Orengo, and Burkhard Rost. Contrastive learning on protein embeddings enlightens midnight zone. *NAR genomics and bioinformatics*, 4(2):lqac043, 2022.
- [413] Maxwell L Bileschi, David Belanger, Drew H Bryant, Theo Sanderson, Brandon Carter, D Sculley, Alex Bateman, Mark A DePristo, and Lucy J Colwell. Using deep learning to annotate the protein universe. *Nature Biotechnology*, 40(6):932–937, 2022.
- [414] Maxat Kulmanov and Robert Hoehndorf. Deepgoplus: improved protein function prediction from sequence. *Bioinformatics*, 36(2):422–429, 2020.
- [415] Fuhao Zhang, Hong Song, Min Zeng, Yaohang Li, Lukasz Kurgan, and Min Li. Deepfunc: a deep learning framework for accurate prediction of protein functions from protein sequences and interactions. *Proteomics*, 19(12):1900019, 2019.
- [416] Maria Littmann, Michael Heinzinger, Christian Dallago, Tobias Olenyi, and Burkhard Rost. Embeddings from deep learning transfer go annotations beyond homology. *Scientific reports*, 11(1):1160, 2021.
- [417] Yue Cao and Yang Shen. Tale: Transformer-based protein function annotation with joint sequence–label embedding. *Bioinformatics*, 37(18):2825–2833, 2021.
- [418] Jia-Ying Chen, Jing-Fu Wang, Yue Hu, Xin-Hui Li, Yu-Rong Qian, and Chao-Lin Song. Evaluating the advancements in protein language models for encoding strategies in protein function prediction: a comprehensive review. *Frontiers in bioengineering and biotechnology*, 13:1506508, 2025.
- [419] Somaye Hashemifar, Behnam Neyshabur, Aly A Khan, and Jinbo Xu. Predicting protein–protein interactions through sequence-based deep learning. *Bioinformatics*, 34(17):i802–i810, 2018.
- [420] Kanchan Jha, Sriparna Saha, and Hiteshi Singh. Prediction of protein–protein interaction using graph neural networks. *Scientific Reports*, 12(1):8360, 2022.
- [421] Nadav Brandes, Grant Goldman, Charlotte H Wang, Chun Jimmie Ye, and Vasilis Ntranos. Genome-wide prediction of disease variant effects with a deep protein language model. *Nature Genetics*, 55(9):1512–1522, 2023.
- [422] Vikas Pejaver, Jorge Urresti, Jose Lugo-Martinez, Kymberleigh A Pagel, Guan Ning Lin, Hyun-Jun Nam, Matthew Mort, David N Cooper, Jonathan Sebat, Lilia M Iakoucheva, et al. Inferring the molecular and

- phenotypic impact of amino acid variants with mutpred2. *Nature communications*, 11(1):5918, 2020.
- [423] Hannes Stärk, Christian Dallago, Michael Heinzinger, and Burkhard Rost. Light attention predicts protein location from the language of life. *Bioinformatics Advances*, 1(1):vbab035, 2021.
- [424] Vineet Thummuluri, José Juan Almagro Armenteros, Alexander Rosenberg Johansen, Henrik Nielsen, and Ole Winther. Deeploc 2.0: multi-label subcellular localization prediction using protein language models. *Nucleic acids research*, 50(W1):W228–W234, 2022.
- [425] Yijia Xiao, Wanjia Zhao, Junkai Zhang, Yiqiao Jin, Han Zhang, Zhicheng Ren, Renliang Sun, Haixin Wang, Guancheng Wan, Pan Lu, et al. Protein large language models: A comprehensive survey. *arXiv preprint arXiv:2502.17504*, 2025.
- [426] Tristan Bepler and Bonnie Berger. Learning the protein language: Evolution, structure, and function. *Cell systems*, 12(6):654–669, 2021.
- [427] Maxat Kulmanov, Mohammed Asif Khan, and Robert Hoehndorf. Deepgo: predicting protein functions from sequence and interactions using a deep ontology-aware classifier. *Bioinformatics*, 34(4):660–668, 2018.
- [428] Ronghui You, Zihan Zhang, Yi Xiong, Fengzhu Sun, Hiroshi Mamitsuka, and Shanfeng Zhu. Golabeler: improving sequence-based large-scale protein function prediction by learning to rank. *Bioinformatics*, 34(14):2465–2473, 2018.
- [429] Liyuan Zhang, Yongquan Jiang, and Yan Yang. Gnn3d: Protein function prediction based on 3d structure and functional hierarchy learning. *IEEE Transactions on Knowledge and Data Engineering*, 36(8):3867–3878, 2023.
- [430] Chenguang Zhao, Tong Liu, and Zheng Wang. Panda-3d: protein function prediction based on alphafold models. *NAR Genomics and Bioinformatics*, 6(3):lqae094, 2024.
- [431] Boqiao Lai and Jinbo Xu. Accurate protein function prediction via graph attention networks with predicted structure information. *Briefings in Bioinformatics*, 23(1):bbab502, 2022.
- [432] Alexander Thomas Julian, Anne Caroline Mascarenhas dos Santos, and Jean-François Pombert. 3dfi: a pipeline to infer protein function using structural homology. *Bioinformatics advances*, 1(1):vbab030, 2021.
- [433] Samuel Sledzieski, Kapil Devkota, Rohit Singh, Lenore Cowen, and Bonnie Berger. Tt3d: Leveraging precomputed protein 3d sequence models to predict protein–protein interactions. *Bioinformatics*, 39(11):btad663, 2023.
- [434] Michael Heinzinger, Konstantin Weissenow, Joaquin Gomez Sanchez, Adrian Henkel, Milot Mirdita, Martin Steinegger, and Burkhard Rost. Bilingual language model for protein sequence and structure. *NAR Genomics and Bioinformatics*, 6(4):lqae150, 2024.

- [435] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140):1–67, 2020.
- [436] Aric Hagberg, Pieter J Swart, and Daniel A Schult. Exploring network structure, dynamics, and function using networkx. Technical report, Los Alamos National Laboratory (LANL), Los Alamos, NM (United States), 2008.
- [437] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. Huggingface’s transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*, 2019.
- [438] Charles R Harris, K Jarrod Millman, Stéfan J Van Der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J Smith, et al. Array programming with numpy. *Nature*, 585(7825):357–362, 2020.
- [439] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. {TensorFlow}: a system for {Large-Scale} machine learning. In *12th USENIX symposium on operating systems design and implementation (OSDI 16)*, pages 265–283, 2016.
- [440] François Chollet et al. Keras. <https://keras.io>, 2015.
- [441] Diederik P Kingma. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.
- [442] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. Isolation-based anomaly detection. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 6(1):1–39, 2012.
- [443] Anant B Parekh and James W Putney Jr. Store-operated calcium channels. *Physiological reviews*, 85(2):757–810, 2005.
- [444] UniProt Consortium et al. Uniprot: the universal protein knowledgebase. *Nucleic acids research*, 46(5):2699, 2018.
- [445] Shuichi Kawashima and Minoru Kanehisa. Aaindex: amino acid index database. *Nucleic acids research*, 28(1):374–374, 2000.
- [446] John Lonsdale, Jeffrey Thomas, Mike Salvatore, Rebecca Phillips, Edmund Lo, Saboor Shad, Richard Hasz, Gary Walters, Fernando Garcia, Nancy Young, et al. The genotype-tissue expression (gtex) project. *Nature genetics*, 45(6):580, 2013.
- [447] Ron Edgar, Michael Domrachev, and Alex E Lash. Gene expression omnibus: Ncbi gene expression and hybridization array data repository. *Nucleic acids research*, 30(1):207–210, 2002.
- [448] Milton H Saier. A functional-phylogenetic classification system for transmembrane solute transporters. *Microbiology and Molecular Biology Reviews*, 64(2):354–411, 2000.

- [449] Catia Pesquita, Daniel Faria, Andre O Falcao, Phillip Lord, and Francisco M Couto. Semantic similarity in biomedical ontologies. *PLoS computational biology*, 5(7):e1000443, 2009.
- [450] Sarah Burge, Elizabeth Kelly, David Lonsdale, Prudence Mutowo-Muellenet, Craig McAnulla, Alex Mitchell, Amaia Sangrador-Vegas, Siew-Yit Yong, Nicola Mulder, and Sarah Hunter. Manual go annotation of predictive protein signatures: the interpro approach to go curation. *Database*, 2012:bar068, 2012.
- [451] Kunjie Fan, Yuanfang Guan, and Yan Zhang. Graph2go: a multi-modal attributed network embedding method for inferring protein functions. *GigaScience*, 9(8):giaa081, 2020.
- [452] Jiawei Wang, Bingjiao Yang, Jerico Revote, Andre Leier, Tatiana T Marquez-Lago, Geoffrey Webb, Jiangning Song, Kuo-Chen Chou, and Trevor Lithgow. Possum: a bioinformatics toolkit for generating numerical sequence feature descriptors based on pssm profiles. *Bioinformatics*, 33(17):2756–2758, 2017.
- [453] Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. *Advances in neural information processing systems*, 30, 2017.
- [454] Adam Santoro, David Raposo, David G Barrett, Mateusz Malinowski, Razvan Pascanu, Peter Battaglia, and Timothy Lillicrap. A simple neural network module for relational reasoning. *Advances in neural information processing systems*, 30, 2017.

Appendix A

Supplementary Material for Chapter 2

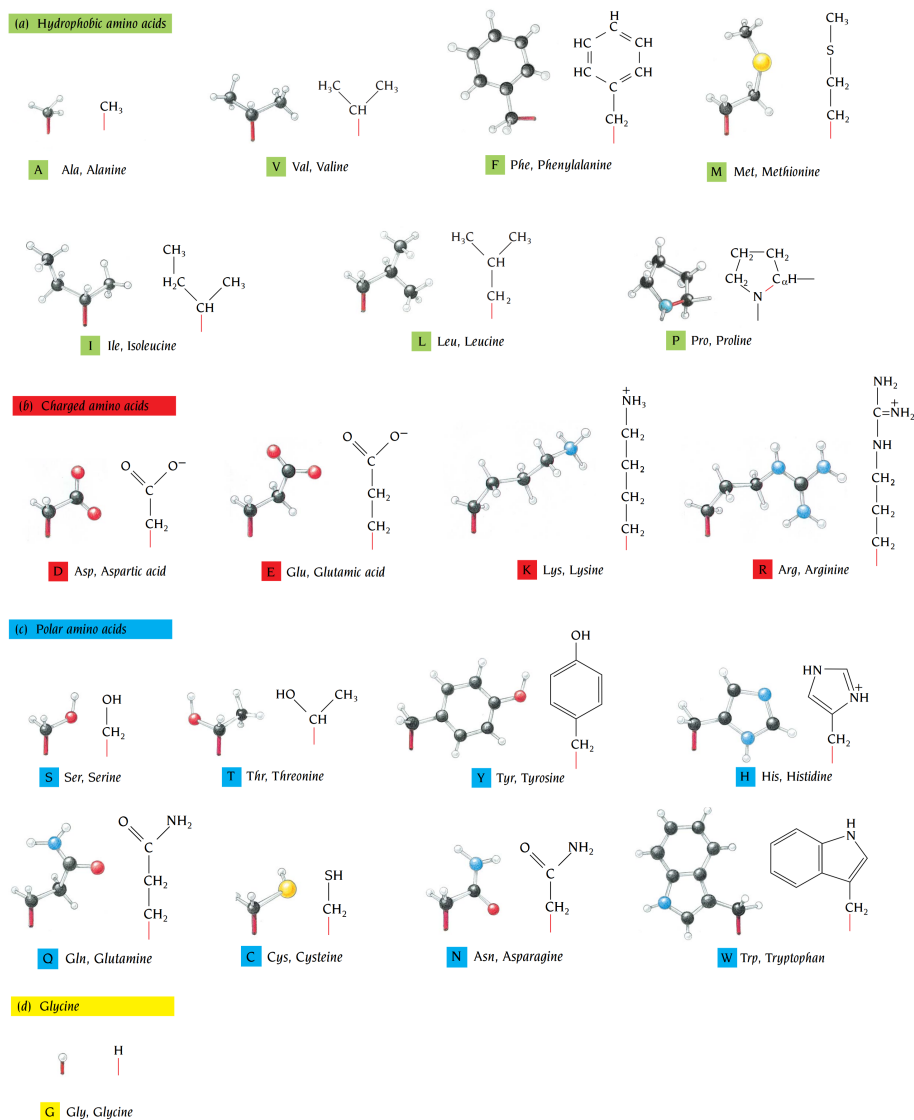


Figure A.1: Amino acid residues found in the 20 proteinogenic amino acids that are encoded by the standard genetic code. Each residue is attached to the C_{α} atom of a L - α -amino acid through the covalent bond colored in red (see Figure 2.1). Adapted from Branden and Tooze [58].

sp Q7KQJ5 HXT1_PLAF7	100.00%	23.28%	25.74%	25.27%	27.72%	23.35%	25.76%	28.31%
sp Q9LT15 STP10_ARATH	23.28%	100.00%	26.90%	26.35%	24.53%	24.60%	28.60%	31.75%
sp P11168 GTR2_HUMAN	25.74%	26.90%	100.00%	52.04%	55.83%	54.25%	27.83%	30.84%
sp P11169 GTR3_HUMAN	25.27%	26.35%	52.04%	100.00%	63.60%	57.64%	29.28%	29.48%
sp P11166 GTR1_HUMAN	27.72%	24.53%	55.83%	63.60%	100.00%	65.45%	28.91%	30.91%
sp P14672 GLUT4_HUMAN	23.35%	24.60%	54.25%	57.64%	65.45%	100.00%	26.25%	29.71%
sp P0AGF4 XYLE_ECOLI	25.76%	28.60%	27.83%	29.28%	28.91%	26.25%	100.00%	31.45%
sp A0A0H2VG78 GLCP_STAES	28.31%	31.75%	30.84%	29.48%	30.91%	29.71%	31.45%	100.00%

Figure A.2: Pairwise sequence identity matrix featuring human GLUT1-4 (P11166, P11168, P11169 and P14672, respectively) and four homologues from *A. thaliana*, *E. coli*, *P. falciparum* and *S. epidermidis* (Q9LT15, P0AGF4, Q7KQJ5 and A0A0H2VG78, respectively). The scores were created using Clustal 2.1 on the UniprotKB, version 2025.01. Despite low sequence identity, these proteins still show high structural conservation, and closely matching binding pockets.

Figure A.3: Multiple sequence alignment (MSA) of human GLUT1-4 and homologues from *A. thaliana*, *E. coli*, *P. falciparum* and *S. epidermidis* (P11166, P11168, P11169, P14672, Q9LTI5, P0AGF4, Q7KWJ5 and A0A0H2VG78, respectively). The MSA was created using Clustal Omega [69] version 1.2.4 on UniprotKB version 2025.01. Known substrate binding sites in human GLUT3 are marked with the letter B. Conservation is shown with the symbols “**”, “.” and “ ” for full, strong and weak conservation, respectively.

```

sp|P11169|GTR3_HUMAN      MGTQKVTPALIFAITVATIGSFQFGYNTGVINAPEKIIKEFINKTLTDKGNAPPSEVLLT 60
sp|P32037|GTR3_MOUSE      MGTTKVTPSLVFAVTVATIGSFQFGYNTGVINAPETILKDFLNYTLERLEDLPSEGLLT 60
*** ***:**:*:*****:***:*:*:* ** : : *** **

sp|P11169|GTR3_HUMAN      SLWSLSVAIFSVMIGSFSVGLFVNRFGRRNSMLIVNLLAVTGGCFMGLCKVAKSVEML 120
sp|P32037|GTR3_MOUSE      ALWSLCVAIFSVMIGSFSVGLFVNRFGRRNSMLLVNLLAIAGCLMGFAKIAESVEML 120
:***:*****:*****:*****: **:*:*:*:*:*****

sp|P11169|GTR3_HUMAN      ILGRLVIGLFCGLCTGFVPMYIGEISPTALRGAFGTNLQGLIVVGILVAQIFGLEFILGS 180
sp|P32037|GTR3_MOUSE      ILGRLLIGIFCGLCTGFVPMYIGVSPALRGAFGTNLQGLIVVGILVAQIFGLDFILGS 180
*****:**:*****:*****:*****:*****

sp|P11169|GTR3_HUMAN      EELWPLLGLFTILPAILQSAALPFCPESPRFLLINRKEENAKQILQRLWGTQDVSQDIQ 240
sp|P32037|GTR3_MOUSE      EELWPGLGLTIIPAILQSAALPFCPESPRFLLINKKEEQATEILQRLWGTSDVQVEIQ 240
***** **:*:*:*****:*****:***:;*:*:***** ** **

sp|P11169|GTR3_HUMAN      EMKDESARMSQEKQVTVLELFRVSSYRQPIIISIVLQLSQQLSGINAVFYYSTGIFKDG 300
sp|P32037|GTR3_MOUSE      EMKDESVRMSQEKQVTVLELFRSPNYVQPLLISIVLQLSQQLSGINAVFYYSTGIFKDG 300
*****:***** ** **:*:*****:*****:*****

sp|P11169|GTR3_HUMAN      VQEPIYATIGAGVVNTIFTVVSLFLVERAGRRTLHMI GLGMAFCSTLMTVSLLLKDNYN 360
sp|P32037|GTR3_MOUSE      VQEPIYATIGAGVVNTIFTVVSLFLVERAGRRTLHMI GLGMAVCSVFM TISLLKDDYE 360
*****:***** **.*:*:*****:*:

sp|P11169|GTR3_HUMAN      GMSFVCIGAILVFVAFFEIGPGPIPWFI VAE LFSQGPRPAAMAVAGCSNWSNFLVGLLF 420
sp|P32037|GTR3_MOUSE      AMSFVCIVAILIYVAFFEIGPGPIPWFI VAE LFSQGPRPAAI VAGCCNWSNFLVGMLF 420
.***** ***:*****:*****:*****:*****:*****

sp|P11169|GTR3_HUMAN      PSAAHYLGAYVFIIFTGFLITFLAFTFFKVPETRGRTFEDITRAFEGQAHSKGDGV 480
sp|P32037|GTR3_MOUSE      PSAAAYLGAYVFIIFAAFLIFFLIFTFFKVPETKGRTFEDIARAFEGQAHSKGPA--G 477
**** *****:*** ** *****:*****:*****:..

sp|P11169|GTR3_HUMAN      MEMNSIEPAKETTTNV 496
sp|P32037|GTR3_MOUSE      VELNSMQPVKETPGNA 493
:*:*:*:*:*** *.

```

Figure A.4: Pairwise sequence alignment between GLUT3 from human (P11169) and mouse (P32037, *mus musculus*). Created with sequence data from UniprotKB version 2025_01; the alignment was calculated with Clustal Omega [69] version 1.2.4. The symbols underneath show the conservation, with "*" for identical positions, ":" for very similar amino acids (strong conservation), "." for weakly conserved positions, and a blank space for dissimilar amino acids, as well as for insertions and deletions.

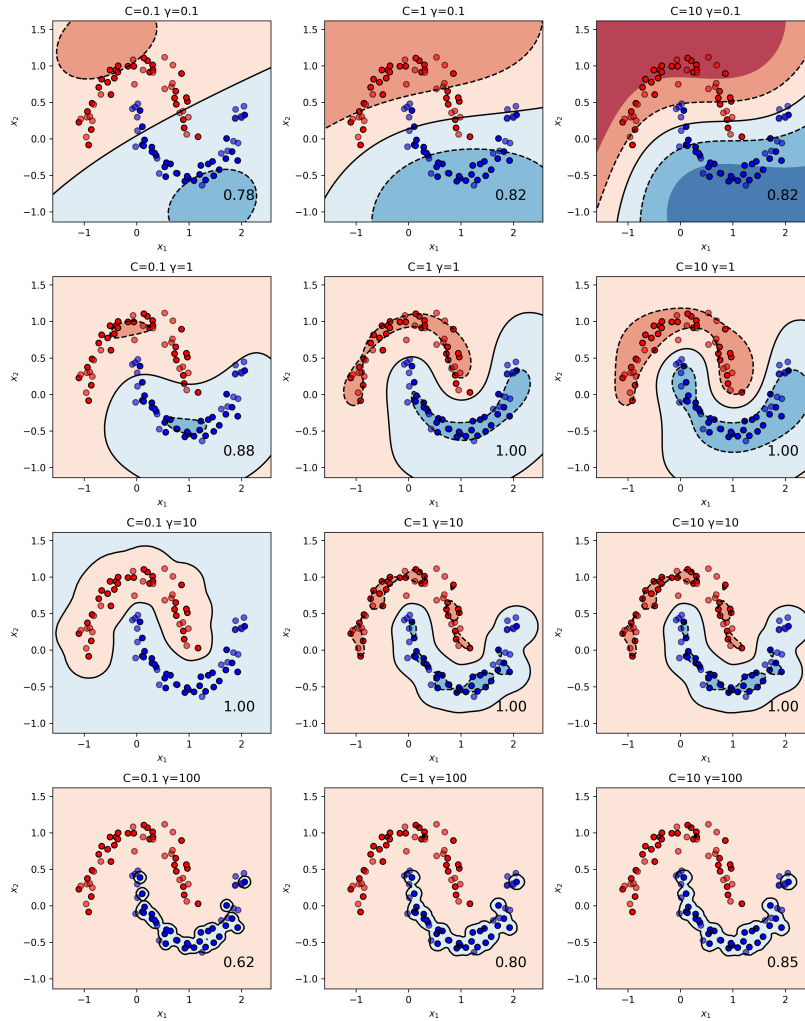


Figure A.5: Nine SVMs with radial basis function (RBF) kernel, with three different values for each of the two regularization parameters γ and C . Otherwise, the models are identical. They were all trained on the same moons dataset, which is not linearly separable in $\mathbb{R}^2$. The background color changes its hue whenever the decision function reaches the next integer. Negative values of the decision function, as well as the negative class, are colored in red. The margin-defining hyperplanes, located at the point where the decision function f reaches 1 and -1, are marked with dashed lines, while the decision-making hyperplane is marked by a straight line. The dataset was split into a training and a test set, in a stratified fashion, meaning that the original fractions of positive and negative samples were kept in each subset. Both the training and the test data was standardized with a z-score, which was derived from the mean and standard deviation of the test set. Samples from the test set are shown in brighter colors, while training samples are colored in a darker shade. The accuracy score of the classifier on the test set is written in the bottom-right of each plot. Lower values of C can reduce overfitting by allowing more samples to enter the margin, while smaller values of gamma regularize the model by limiting the influence of each individual sample on the decision function, which leads to a smoother decision boundary [325]. Low values of each parameter can lead to problems with under-fitting, which can be seen especially in the top row, while the overfitting models in the bottom row are caused by a high γ -value. Dataset creation, machine learning and visualization were carried out with scikit-learn [56] and matplotlib [52].

Appendix B

Supplementary Material for Chapter 3

B.1 Code and raw data

The raw data, along with all the code that was implemented, is available for download at <https://github.com/adenger/subpred3>.

B.2 Uniprot Keywords

Substrate keywords

Ion transport, Anion exchange, Protein transport, Sodium/potassium transport, Polysaccharide transport, Bacteriocin transport, Peptide transport, Translocation, Bacterial flagellum protein export, Amino-acid transport, Electron transport, Lipid transport, mRNA transport, Neurotransmitter transport, Oxygen transport, Phosphate transport, Ammonia transport, Phosphonate transport, Viral movement protein, Sulfate transport, Sugar transport, Calcium transport, Cobalt transport, Copper transport, Hydrogen ion transport, Iron transport, Zinc transport, Nickel transport, Potassium transport, Sodium transport, Chloride

Location keywords

Membrane, Cell membrane, Cell inner membrane, Cell outer membrane, Transmembrane, Nucleus, Mitochondrion, Endoplasmic reticulum, Plastid outer membrane, Plastid inner membrane, Mitochondrion outer membrane, Mitochondrion inner membrane, Postsynaptic cell membrane

Other transport-related keywords

Antibiotic resistance, Transport, Symport, Antiport, ER-Golgi transport, Ion channel, Calcium channel, Potassium channel, Chloride channel, Sodium chan-

nel, Viral ion channel, Voltage-gated channel, Ligand-gated ion channel, Porin,
Nuclear pore complex, Respiratory chain

Appendix C

Supplementary Material for Chapter 4

C.1 Code and raw data

The raw data, along with all the code that was implemented, is available for download at <https://github.com/adenger/subpred4>.

C.2 Steps in the algorithm pipeline

Input

- **T** Transport protein dataset and corresponding GO annotation data created with the transporter dataset creation pipeline described in Section 4.3.2
- **O** Ontology graph for the annotation data in OBO format, created by the preprocessing pipeline (see Section 4.3.1)

Parameters

- **n** GO terms annotated with fewer than n proteins in the dataset are removed. (Default: 20)
- **m** GO terms annotated with fewer than m unique proteins are removed, i.e. proteins that are only annotated with one of the GO terms in a binary classification task, not both. (Default: 15)
- **p** The top p th percentile of GO terms according to the distribution of annotated samples is removed (see Section 4.3.4). (Default: 0)
- **c** The minimum allowed coverage of the resulting set of GO terms, i.e. the percentage of proteins in the original protein dataset that are annotated with a GO term from the optimized subset. (Default: 0.9)
- **a** Whether to prefer abstract or specific GO terms. Higher specificity is defined as increased distance to root node. (Default: specific)

- ϵ Reduces the influence of rounding errors and local minima on the algorithm, by allowing the optimization to consider GO terms that are within ϵ of the best score during the iteration. Scores are in the interval $[0, 1]$, therefore values of $\epsilon \leq 0.05$ are recommended. (Default: 0.0)

Steps

1. Create a starting set of GO terms to optimize from the provided proteins and their annotations.
2. Filter the GO terms in the set, as well as the corresponding ontology graph, according to parameters n , m and p .
3. Train and evaluate pairwise binary ML models for GO terms, as described in Section 4.3.3.5
4. For each GO term calculate:
 - (a) Coverage loss: The percentage of protein that would be annotated if the GO term was removed from the current set.
 - (b) Score delta: The average ML evaluation score across all pairwise models, if the GO term was removed from the current set.
5. Removal candidates: Select all GO terms that can be removed without the coverage falling below c .
6. Sort removal candidates by score delta, select those with the highest improvement in average F1 score when removed, while also considering those that are within ϵ of the maximum score.
7. End condition: If the list of removal candidates is empty then return the optimized list of GO terms.
8. If the list of removal candidates contains more than one GO term:
 - (a) Select either the most specific or most abstract term(s), depending on parameter a .
 - (b) If the list of removal candidates still contains more than one GO term, draw one of them using deterministic random sampling.
9. Remove the selected GO term from the set, go back to step 4 and repeat the process with the smaller subset.

C.3 Parameters of the transporter dataset pipeline

Manually curated proteins Proteins that have not been manually curated (i.e. those available in TrEMBL but not in SwissProt) can be removed from the dataset.

Sequence evidence Proteins with experimental sequence evidence at transcript level but not at protein level can be added to the dataset. Predicted sequences are removed during pre-processing, and are not available for the pipeline.

GO evidence GO annotations can be filtered for their evidence codes, e.g. to remove electronically inferred information that has not been manually reviewed. Annotations with the NOT prefix, which denotes that a protein is explicitly *not* annotated with a particular GO term, are already removed during pre-processing.

Organisms Proteins can be filtered for a subset of taxonomy identifiers, for example “9606” for human.

External proteins Proteins that do not occur in the provided organisms can be added to the dataset, e.g. to simulate experiments where proteins from one organism are expressed in another one.

Gene names available The 671,750 proteins without any annotated gene names can be removed from the dataset, for example if the data needs to be linked to genetics- or transcriptomics datasets.

GO subset A sub-tree of GO can be selected by providing a new root node. Further filtering can be applied to the GO subset, for example for the aspect (i.e. *molecular_function*, *cellular_component*, *biological_process*) and for a subset of edge annotations (relationships) between GO terms (e.g. *is_a*). All GO terms that are not descendants of the new root node, according to the filtered subset of edge annotations, are then removed from the graph. Only proteins annotated with this subset of GO terms remain in the dataset.

ChEBI quality ChEBI terms can optionally be filtered by their star rating system, e.g. for those that have been manually verified.

Cellular component Optionally, a subset of cellular component annotations can be created, which then get filtered for those with the *located_in* or *is_active_in* qualifiers, and only *is_a* relations between GO terms are kept. Proteins are also annotated with the ancestor GO terms of their cellular components. The subset of transport proteins that are located in a particular set of compartments, such as *plasma membrane*, *mitochondrion* or *eisosome*, can then be selected for further analysis.

Swissprot reviewed	Known gene	GO evidence	Protein evidence	Cluster thresh.[%]	Proteins	
					Four organisms	Uniprot
True	True	experiment	protein level	50	1313	1331
				70	1602	1852
				90	1779	2678
				100	1892	3919
				None	1915	4022
			transcript level	50	270	449
				70	384	808
				90	487	1385
				100	521	2092
				None	532	2147
		computational	protein level	50	1185	1649
				70	1492	2391
				90	1664	3415
				100	1793	4824
				None	1821	4990
			transcript level	50	280	585
				70	426	1087
				90	554	1880
				100	592	2868
				None	607	2989
False	True	experiment	protein level	50	24	392
				70	26	573
				90	31	841
				100	34	1279
				None	34	1374
			transcript level	50	1	430
				70	2	813
				90	3	1424
				100	6	2344
				None	7	2538
		computational	protein level	50	188	905
				70	262	1426
				90	456	2211
				100	1080	4373
				None	1294	5016
			transcript level	50	44	3263
				70	65	6228
				90	119	11625
				100	538	24217
				None	789	27705
	False	experiment	transcript level	50	1	18
				70	1	37
				90	1	81
				100	2	116
				None	3	117
		computational	protein level	50	1	19
				70	1	30
				90	5	58
				100	9	181
				None	9	185
			transcript level	50	61	3820
				70	77	7133
				90	145	12032
				100	648	19559
				None	751	21208

Table C.1: Number of unique proteins in the transmembrane transporter dataset, after filtering for criteria related to data quality. The proteins were filtered for those annotated with the *transmembrane transporter activity* GO term. Sequence clustering was carried out with cd-hit [317].

Swissprot reviewed	Known gene	GO evidence	Protein evidence	Cluster thresh.[%]	Transporter GO terms	
					Four organisms	Uniprot
True	True	experiment	protein level	50	650	581
				70	679	639
				90	693	711
				100	702	746
				None	705	747
			transcript level	50	268	349
				70	300	420
				90	314	480
				100	319	540
				None	322	542
		computational	protein level	50	449	390
				70	492	462
				90	499	546
				100	509	616
				None	509	617
			transcript level	50	166	232
				70	189	282
				90	197	337
				100	203	440
				None	205	443
False	True	experiment	protein level	50	62	299
				70	64	320
				90	67	347
				100	70	388
				None	70	389
			transcript level	50	17	261
				70	19	336
				90	22	372
				100	30	397
				None	43	400
		computational	protein level	50	157	261
				70	176	299
				90	188	360
				100	207	441
				None	213	447
			transcript level	50	98	254
				70	101	311
				90	112	373
				100	147	502
				None	195	520
	False	experiment	transcript level	50	8	56
				70	8	90
				90	8	131
				100	8	153
				None	23	153
		computational	protein level	50	5	32
				70	5	41
				90	28	64
				100	28	80
				None	28	80
			transcript level	50	118	215
				70	119	247
				90	123	277
				100	186	302
				None	196	306

Table C.2: Number of unique GO terms in the transmembrane transporter dataset, after filtering for criteria related to data quality. The proteins were filtered for those annotated with the *transmembrane transporter activity* GO term. Sequence clustering was carried out with *cd-hit* [317].

			min	mean	median	max	sdev
Clustering	Dataset	Method					
None	train	None	0.11	0.86	0.90	0.99	0.14
		ANOVA	0.41	0.88	0.91	1.00	0.11
		PCA	0.37	0.87	0.90	1.00	0.10
	test	None	0.15	0.87	0.91	0.99	0.13
		ANOVA	0.37	0.88	0.90	1.00	0.11
		PCA	0.23	0.87	0.90	1.00	0.12
70%	train	None	0.12	0.83	0.87	0.99	0.14
		ANOVA	0.45	0.85	0.88	0.99	0.11
		PCA	0.38	0.84	0.87	0.97	0.11
	test	None	0.13	0.84	0.88	0.98	0.14
		ANOVA	0.42	0.84	0.87	0.98	0.11
		PCA	0.26	0.84	0.87	0.98	0.11

Table C.3: F1 scores for different machine learning models that were trained on pairs of GO terms and their associated proteins, optionally with 70% sequence clustering, for the yeast dataset. The goal was to remove the worst-performing GO terms with the clustering pipeline described in Section 4.3.4. ANOVA and PCA refer to two different types methods that were used to reduce the number of feature dimensions (see Section 4.3.3.5)

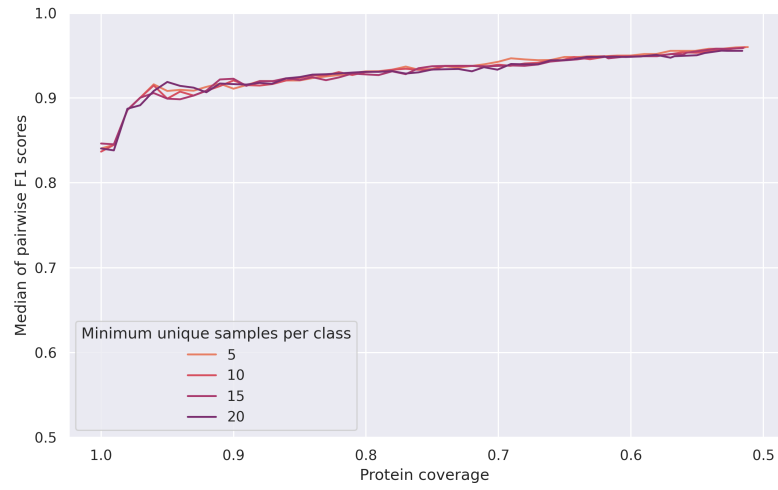


Figure C.1: Median F1 scores between pairs of GO terms for the meta-organism dataset, when not removing the top 5th percentile of GO terms according to sample count.



Figure C.2: Final subset sizes for the meta-organism dataset after applying the pipeline, when not removing the top 5th percentile of GO terms according to sample count.

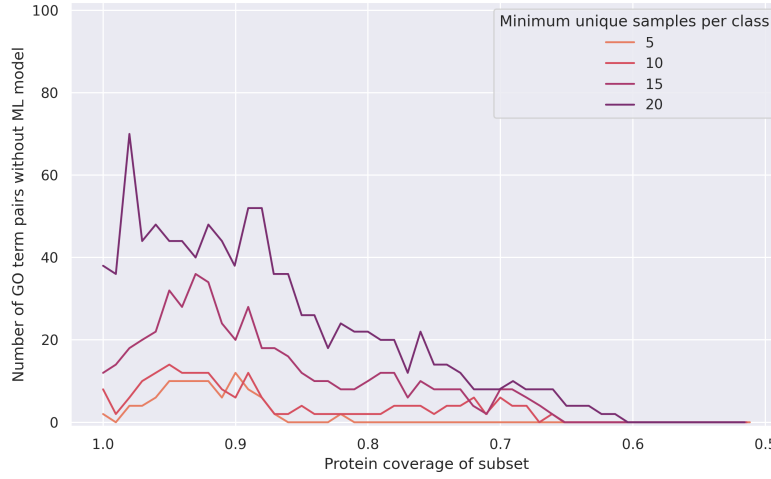


Figure C.3: Number of GO terms with no available $F1$ scores for the meta-organism dataset, when not removing the top 5th percentile of GO terms according to sample count.

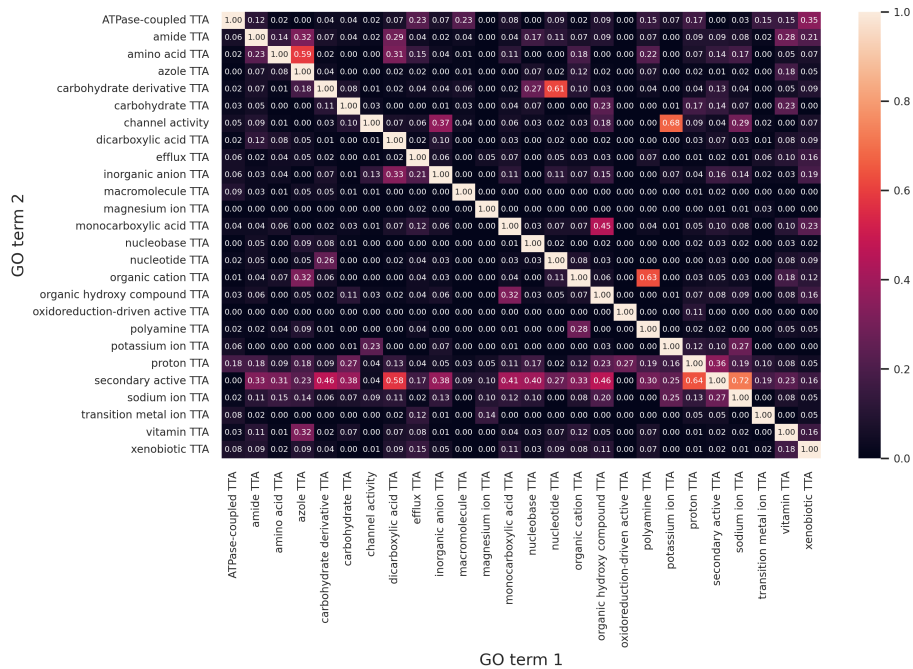


Figure C.4: Heatmap showing the fraction of proteins annotated with GO term 1 that are also annotated with GO term 2, for all pairs of GO terms in the optimized subset for the meta-organism dataset, but without removing the top 5th percentile of GO terms.

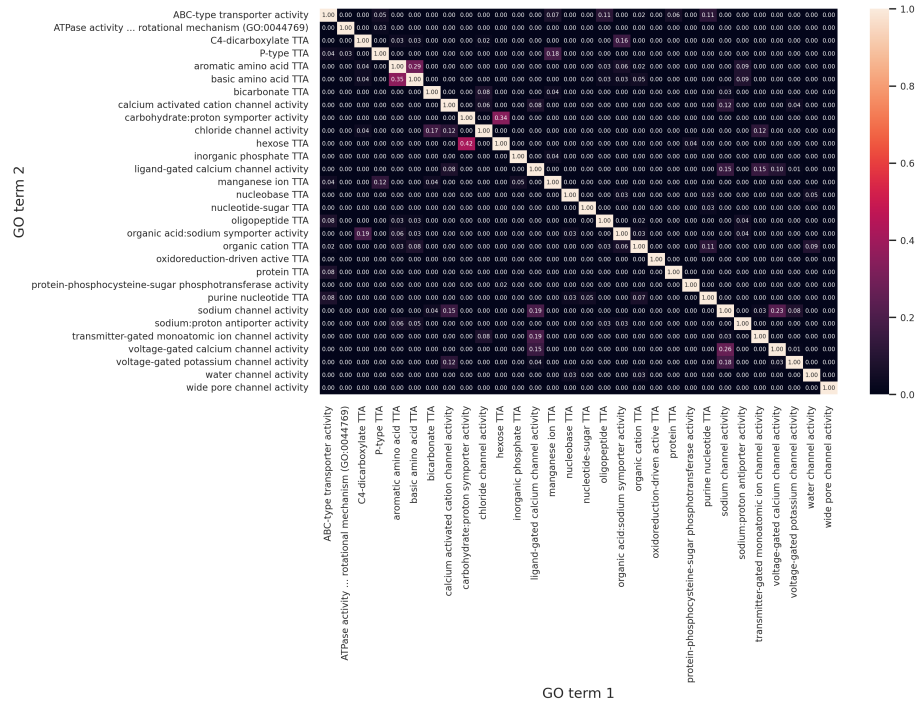


Figure C.5: Heatmap showing of the fraction of proteins annotated with GO term 1 that is also annotated with GO term 2, for all pairs of GO terms in the optimized subset for the meta-organism dataset, when removing the top 5th percentile of GO terms. Here, the coverage threshold was reduced to 67, and m was set to 20.

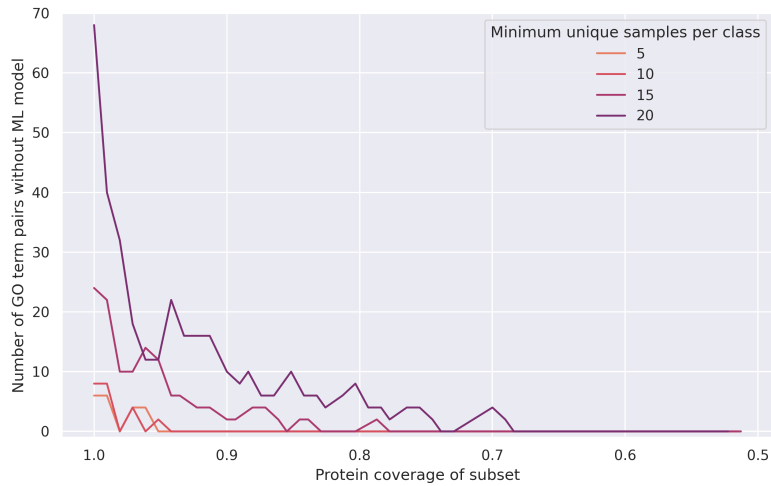


Figure C.6: Number of GO term pairs in the yeast dataset without evaluation scores available, at different protein coverage thresholds and for four different values of m . ML models are only available for pairs that are distinct enough, meaning that each term has at least m proteins available for training that are not also annotated with the respective other term.

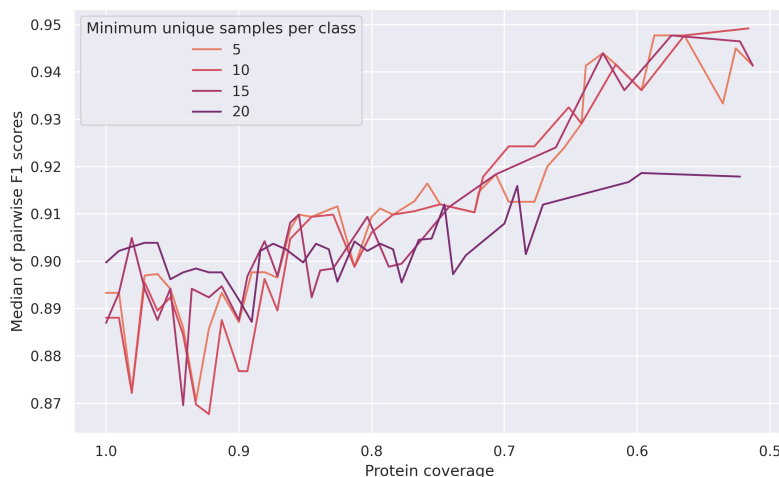


Figure C.7: Median F1 scores between pairs of GO terms in the yeast dataset, at different protein coverage thresholds, using four different evaluation matrices that were created with different values of m , i.e. the threshold for how few unique samples are allowed per class during training.

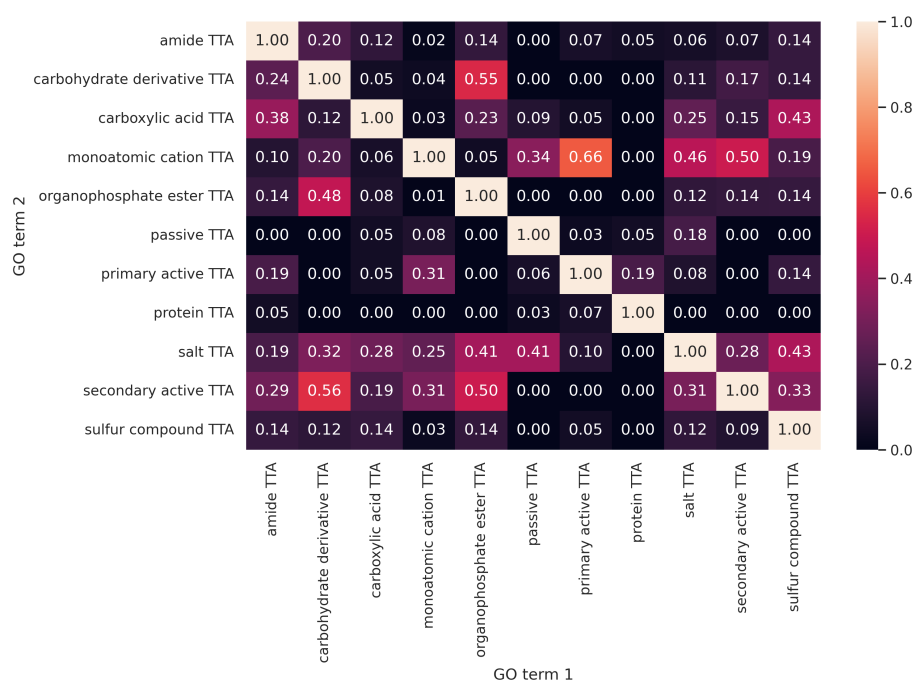


Figure C.8: Heatmap showing the fraction of yeast dataset proteins annotated with GO term 1 that are also annotated with GO term 2, for all pairs of GO terms in the optimized subset. Removal of any term would cause the protein coverage to fall below the specified 98%.

Appendix D

Supplementary Material for Chapter 5

Feature	Balanced Accuracy	F1 Amino Acid	F1 Sugar
DUMMY	0.463 ± 0.122	0.393 ± 0.171	0.509 ± 0.118
AAC	0.711 ± 0.159	0.705 ± 0.152	0.700 ± 0.202
PAAC	0.892 ± 0.087	0.893 ± 0.086	0.881 ± 0.099
AA-KMER3	0.883 ± 0.082	0.883 ± 0.080	0.870 ± 0.099
MULTI-PSSM	0.816 ± 0.122	0.815 ± 0.118	0.799 ± 0.153
META	0.818 ± 0.134	0.813 ± 0.139	0.808 ± 0.153
META-STD	0.869 ± 0.125	0.866 ± 0.129	0.864 ± 0.128
COMB-KMER1	0.897 ± 0.082	0.894 ± 0.084	0.894 ± 0.088
COMB-KMER2	0.907 ± 0.079	0.908 ± 0.076	0.898 ± 0.092
COMB-KMER3	0.947 ± 0.072	0.947 ± 0.072	0.942 ± 0.078
PROTT5-AA	0.910 ± 0.058	0.900 ± 0.070	0.917 ± 0.049
PROSTT5-AA	0.913 ± 0.079	0.902 ± 0.102	0.921 ± 0.064
PROSTT5-3Di	0.909 ± 0.074	0.899 ± 0.087	0.917 ± 0.067
3Di-COMP	0.929 ± 0.106	0.917 ± 0.127	0.936 ± 0.091
3Di-KMER2	<u>0.971 ± 0.044</u>	<u>0.969 ± 0.046</u>	<u>0.970 ± 0.045</u>
3Di-KMER3	0.920 ± 0.054	0.916 ± 0.058	0.920 ± 0.055

Table D.1: Evaluation results for each feature after a nested cross-validation that was repeated five times on shuffled data. For each split, an SVM model was trained to distinguish between sugar- amino acid transporters in *A. thaliana*. The scores represent the mean and standard deviation of the independent test sets, across all 25 splits. The underlying protein dataset was automatically generated from Uniprot and annotated with GO terms, in a way that relies on experimental data as much as possible, while filtering out predicted annotations and proteins. The best mean and standard deviation for each column is underlined.

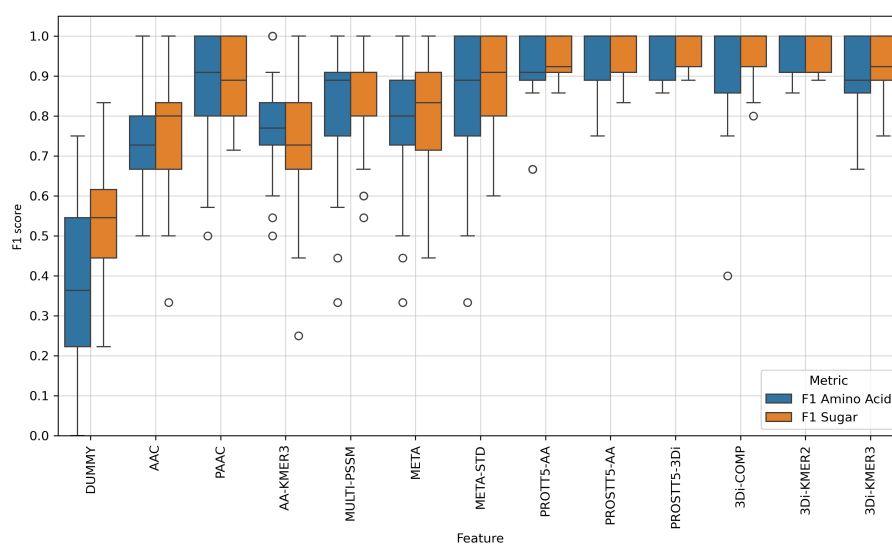


Figure D.1: Evaluation results for an SVM model that can distinguish sugar- from amino acid transporters in *A. thaliana*, after automatic outlier removal. This plot visualizes the distribution of F1 scores for each class, after five shuffled repetitions of a nested 5-fold cross-validation. Each box represents the 25th to 75th percentile, the line in the box is the median, and the length of the whiskers is limited to 1.5 times the distance between the 25th and 75th percentile. Scores outside of that range are shown dots, and classified as outliers. In contrast to Figure 5.1, an automatic outlier detection method has previously been applied to the data, in order to find and remove biological edge cases, and potential errors in the data. As in Figure 5.1, the 3Di-KMER2 feature delivered the best-performing model. The classification performance of all three embeddings was improved by the outlier removal. The consistency of evaluation scores was also improved, with lower standard deviations for several features.

Feature	Balanced Accuracy	F1 Amino Acid	F1 Sugar
DUMMY	0.480 ± 0.136	0.354 ± 0.232	0.534 ± 0.156
AAC	0.766 ± 0.116	0.731 ± 0.139	0.770 ± 0.147
PAAC	0.881 ± 0.105	0.864 ± 0.134	0.879 ± 0.095
AA-KMER3	0.770 ± 0.133	0.774 ± 0.126	0.727 ± 0.171
MULTI-PSSM	0.842 ± 0.138	0.812 ± 0.173	0.847 ± 0.133
META	0.810 ± 0.154	0.783 ± 0.181	0.813 ± 0.153
META-STD	0.869 ± 0.135	0.844 ± 0.176	0.876 ± 0.126
COMB-KMER1	0.951 ± 0.080	0.945 ± 0.091	0.951 ± 0.086
COMB-KMER2	0.938 ± 0.083	0.934 ± 0.086	0.933 ± 0.092
COMB-KMER3	0.914 ± 0.088	0.906 ± 0.097	0.917 ± 0.086
PROTT5-AA	0.921 ± 0.073	0.910 ± 0.092	0.938 ± 0.051
PROSTT5-AA	0.948 ± 0.060	0.942 ± 0.069	0.954 ± 0.052
PROSTT5-3Di	0.949 ± 0.055	0.942 ± 0.062	$0.959 \pm \underline{0.044}$
3Di-COMP	0.934 ± 0.098	0.915 ± 0.140	0.954 ± 0.065
3Di-KMER2	$\underline{0.967} \pm \underline{0.046}$	$\underline{0.961} \pm \underline{0.054}$	$\underline{0.968} \pm \underline{0.044}$
3Di-KMER3	0.899 ± 0.078	0.891 ± 0.088	0.912 ± 0.073

Table D.2: Evaluation results for each feature, after a nested cross-validation that was repeated five times on shuffled data. For each split, a SVM model was trained to distinguish between sugar and amino acid transporters in *A. thaliana*. The scores represent the means and standard deviations of the independent test sets across all 25 splits. The underlying protein dataset was automatically generated from Uniprot and annotated with GO terms, in a way that relies on experimental data as much as possible, while filtering out predicted annotations and proteins. In contrast to Table D.1, an outlier prediction pipeline was applied to the dataset before the evaluation, which removed three proteins. The best mean and standard deviation for each column is underlined.

Feature Name	Data Source	Type	Dimension
DUMMY	numpy.random	Unif. random values	1024
AAC	Amino acid sequence	1-mer frequencies	20
PAAC	Amino acid sequence	2-mer frequencies	400
AA-KMER3	Amino acid sequence	3-mer frequencies	8000
PSSM-50-1	PSIBLAST (Uniref50, 1 iteration)	Evol. information	400
PSSM-50-3	PSIBLAST (Uniref50, 3 iterations)	Evol. information	400
PSSM-90-1	PSIBLAST (Uniref90, 1 iteration)	Evol. information	400
PSSM-90-3	PSIBLAST (Uniref90, 3 iterations)	Evol. information	400
MULTI-PSSM	PSIBLAST (Four PSSMs)	Evol. information	1600
META	AAC + PAAC + MULTI-PSSM	Combined	2020
META-STD	META, standardized samples	Fingerprint	2020
3Di-COMP	3Di sequence	1-mer frequencies	20
3Di-KMER2	3Di sequence	2-mer frequencies	400
3Di-KMER3	3Di sequence	3-mer frequencies	8000
COMB-KMER1	AAC + 3Di-COMP	1-mer frequencies	40
COMB-KMER2	PAAC + 3Di-KMER2	2-mer frequencies	800
COMB-KMER3	AA-KMER3 + 3Di-KMER3	3-mer frequencies	1600
PROTT5-AA	Amino acid sequence	pLM embedding	1024
PROSTT5-AA	Amino acid sequence	pLM embedding	1024
PROSTT5-3Di	3Di sequence	pLM embedding	1024

Table D.3: Overview of machine learning features, along with their input data, the type of information they contain, and feature vector dimensions. Amino acid sequences were retrieved from Uniprot and used to calculate k -mer frequencies as well as PSSMs. The PSSM feature contains evolutionary exchange likelihoods for all 20 amino acid types found in protein sequences, calculated individually for each protein. The Multi-PSSM feature is then created by concatenating the four PSSM features into one vector. 3Di sequences were calculated with FoldSeek on predicted protein structures retrieved from AlphaFoldDB. In the COMB features, k -mers derived from the amino acid sequence and predicted 3D structure are combined. The protein language model (pLM) embeddings were mean-pooled along the sequence in order to turn them into per-protein embeddings. The non-informative DUMMY feature contains random values drawn from a uniform random distribution, and is used to estimate the additional information that is provided by each of the real features calculated from protein data.