DATA PREPROCESSING AND QUALITY DIAGNOSIS IN DEEP LEARNING-BASED IN SILICO BIOACTIVITY PREDICTION

AUTHOR: ÁNGELA LÓPEZ DEL RÍO
ADVISOR: ALEXANDRE PERERA LLUNA

A thesis by compendium of publications
submitted in fulfilment of the requirements
for the degree of Doctor of Philosophy
in Biomedical Engineering

in the

B2SLab
Centre de Recerca en Enginyeria Biomèdica
Departament d'Enginyeria de Sistemes, Automàtica i Informàtica Industrial
Universitat Politècnica de Catalunya

April 2021

ABSTRACT

DATA PREPROCESSING AND QUALITY DIAGNOSIS IN DEEP LEARNING-BASED IN SILICO BIOACTIVITY PREDICTION

ÁNGELA LÓPEZ DEL RÍO

Drug discovery is a very time and resource consuming process which involves the identification of a target and the exploration of suitable drug candidates for it, making sure that these candidates do not bind to other targets related to pathways with unwanted outcomes.

To streamline drug discovery, computational techniques are widely applied. They help identifying most likely molecular candidates, retaining the ones with most desirable pharmacokinetic properties and modeling their interactions with the target. However, these techniques are in constant improvement thanks to the development of more sophisticated algorithms, the rising of hardware computational power and the growth of publicly available molecular and structural databases. Specifically, machine learning approaches fit models on publicly annotated data for biochemical properties and target-ligand binding prediction.

Deep learning is a machine learning approach that applies multilayer neural networks to learning tasks, automatically extracting multiple levels of representations of the data. Within the last ten years, these algorithms have outperformed classical prediction models in most domains thanks to the availability of large amounts of data and the maturity of Graphics Processor Unit-accelerated computing. This kind of approach is a perfect fit for the screening of massive compound databases, thus being increasingly applied to drug discovery applications. Common use cases include molecular property prediction, *de novo* compound generation, protein secondary structure prediction and, especially, target-compound binding prediction.

However, the accuracy of deep learning target-compound binding prediction models has been questioned lately. Different studies point out that their reported performance could be a consequence of data bias rather than generalization capability. Although efforts are being put in order to address this problem, it is still present in many state of the art studies, rewarding novelty over critical assessment. Moreover, the flexibility of deep learning derives in a lack of consensus on how to represent the input spaces, which makes it difficult to compare models in a common benchmark. Bioactivity data availability and quality are limited because of its associated costs and it is often imbalanced, which could also difficult the model learning process. The diagnosis of all these problems is not straightforward, since deep learning models are considered black boxes. This hinders the adoption of these methodologies as the *de facto* solution in computer-aided drug discovery.

The present thesis aims to improve deep learning models for computational drug discovery, focusing in the representation of the input, the control of the data bias, the correction of the data imbalance and the diagnosis of the

models. The analyses performed are framed within the context of sequential representation of proteins, aiming to avoid the loss of information linked to other types of representations.

First, this thesis asseses the effect that different validation strategies have on binding classification models, aiming to find the most realistic performance estimates. The strategy based on clustering molecules to avoid having similar compounds in training and test sets showed to be the most similar to a prospective validation, and thus, more consistent than random cross-validation (over-optimistic) or than an external test set from other database (over-pessimistic).

Second, this thesis focuses on the pre-treatment of sequential inputs, specifically on sequence padding. Padding is necessary for establishing a common length by adding zeros to the sequence. Although zeros are usually added at the end of the sequence, there is no formal justification behind it. Here, classical and novel padding strategies were compared in an enzyme classification task. Results showed that the padding position of amino acid sequences has an effect in the performance of deep learning models, so padding should be tuned as an additional hyperparameter.

Third, this thesis studies the effect of data imbalance in protein-compound activity classification models and the ability of resampling techniques to mitigate it. The model performance was assessed for different combinations of oversampling (of the minority class) and clustering. Results showed that the proportion of actives predicted by the model for a given protein was explained by the actual data balance of that protein in the test set, rather than that of the training set. Data clustering, followed by data resampling in training and validation sets, stood as the best performing strategy without altering the test set.

To accomplish the three points above, this thesis provides a systematic way to diagnose deep learning models, identifying the factors that govern the model predictions and performance. Specifically, explanatory linear models enabled informed, quantitative decisions regarding input preprocessing. This ultimately leads to more consistent and trustable results in deep learning target-compound binding prediction models.

PREPROCESAMIENTO DE DATOS Y DIAGNÓSTICO DE CALIDAD EN LA PREDICCIÓN IN SILICO DE BIOACTIVIDAD BASADA EN APRENDIZAJE PROFUNDO

ÁNGELA LÓPEZ DEL RÍO

El descubrimiento de fármacos es un proceso que consume mucho tiempo y recursos. Consiste en la identificación de una diana y la exploración de posibles fármacos candidatos para ella, asegurando que estos no se unen a otras dianas implicadas en vías biológicas con consecuencias no deseadas.

Para optimizar el descubrimiento de fármacos se aplican técnicas computacionales. Estas técnicas ayudan a identificar las moléculas candidatas más probables, a quedarse con aquellas con las propiedades farmacocinéticas más deseables y a modelar sus interacciones con la diana. Sin embargo, estas técnicas están en constante mejora gracias al desarrollo de algoritmos más sofisticados, al incremento del poder computacional y al crecimiento de bases de datos moleculares y estructurales disponibles de forma pública. Específicamente, las estrategias basadas en aprendizaje automático ajustan modelos en datos anotados públicamente para la predicción de propiedades bioquímicas y de unión entre dianas y ligandos.

El aprendizaje profundo es una aproximación del aprendizaje automático que aplica redes neuronales multicapa en tareas de aprendizaje. Estas redes son capaces de extraer de forma automática múltiples niveles de representación de los datos. Durante los últimos diez años, estos algoritmos han conseguido mejores resultados que los modelos de predicción clásicos en la mayoría de dominios gracias a la disponibilidad de grandes cantidades de datos y a la madurez de la computación acelerada por unidades de procesamiento gráfico. Este tipo de estrategia es perfecta para el cribado masivo de bases de datos de compuestos, y por eso cada vez se aplica más en el descubrimiento de fármacos. Algunos de los casos de uso más comunes son la predicción de propiedades moleculares, la generación de compuestos *de novo*, la predicción de la estructura secundaria de las proteínas y, en especial, la predicción de la unión entre compuestos y dianas.

Sin embargo, en los últimos tiempos se ha cuestionado la precisión de los modelos basados en aprendizaje profundo que predicen la unión entre dianas y compuestos. Diferentes estudios apuntan a que el rendimiento reportado de estos modelos podría deberse más al sesgo de los datos que a su capacidad de generalización. Aunque se están haciendo esfuerzos para solucionar este problema, aún está presente en muchos estudios del estado del arte, dando más peso a la novedad que a la valoración crítica. Además, la flexibilidad del aprendizaje profundo da pie a una falta de consenso a la hora de representar los espacios de entrada, lo cual dificulta la comparación de modelos en un marco común. La disponibilidad y calidad de los datos de bioactividad son limitadas debido a sus costes asociados, y a menudo estos datos están desbalanceados, lo cual podría dificultar el proceso de aprendizaje del modelo. El diagnóstico de estos problemas no es sencillo, dado que los modelos de aprendizaje profundo se consideran cajas negras. Este hecho dificulta la adopción de estas metodologías como la solución *de facto* en el descubrimiento de fármacos asistido por ordenador.

La presente tesis tiene como objetivo mejorar los modelos de aprendizaje profundo para el descubrimiento computacional de fármacos, centrándose en la representación de la entrada, el control del sesgo de los datos, la corrección del desbalance de los mismos y el diagnóstico de los modelos. Los análisis llevados a cabo se enmarcan en el contexto de la representación secuencial de las proteínas y los compuestos, con el objetivo de evitar la pérdida de información asociada a otros tipos de representaciones.

En primer lugar, esta tesis evalúa el efecto que tienen diferentes estrategias de validación en los modelos de clasificación de la unión diana-compuesto, con el objetivo de encontrar las estimaciones de rendimiento más realistas. La estrategia basada en el agrupamiento de las moléculas para evitar tener moléculas similares en los conjuntos de entrenamiento y test demostró ser la más parecida a una validación prospectiva, y por tanto, más consistente que la validación cruzada aleatoria (demasiado optimista) o que un conjunto de test externo proveniente de otra base de datos (demasiado pesimista).

En segundo lugar, esta tesis se centra en el pre-tratamiento de las entradas secuenciales, concretamente en el relleno de secuencias. El relleno es necesario para establecer una longitud común mediante la adición de ceros a la secuencia. Aunque estos ceros se aãden normalmente al final de la secuencia, no hay una justificación formal detrás de esta decisión. Aquí se compararon estrategias de relleno novedosas y clásicas en una tarea de clasificación de enzimas. Los resultados mostraron que la posición del relleno añadido a las secuencias de amino ácidos tiene un efecto sobre el rendimiento de los modelos de aprendizaje profundo, por lo que debería ser afinado como cualquier otro hiperparámetro.

En tercer lugar, esta tesis estudia el efecto del desbalance de los datos en los modelos de clasificación de actividad diana-compuesto, y la habilidad de las técnicas de remuestreo para atenuarlo. Se evaluó el rendimiento de un modelo para diferentes combinaciones de sobremuestreo (de la clase minoritaria) y agrupamiento de las moléculas. Los resultados demostraron que la proporción de activos predicha por el modelo para cierta proteína estaba explicada por el balance de datos real de dicha proteína en el conjunto de test, en lugar del mismo en el conjunto de entrenamiento. El agrupamiento de los datos, seguido por su remuestreo en los conjuntos de entrenamiento y validación, resultó ser la estrategia con mejor rendimiento sin alterar el conjunto de test.

Para llevar a cabo los tres puntos anteriores, esta tesis proporciona una forma sistemática de diagnosticar modelos de aprendizaje profundo, identificando los factores que rigen sus predicciones y su rendimiento. Concretamente, los modelos lineales explicativos posibilitaron la toma de decisiones informadas y cuantitativas relacionadas con el preprocesamiento de la entrada. En consecuencia, esto lleva a resultados más consistentes y fiables en los modelos de clasificación de la unión diana-compuesto basados en aprendizaje profundo.

"Would you tell me, please, which way I ought to go from here?"
"That depends a good deal on where you want to get to," said the Cat.
"I don't much care where–" said Alice.
"Then it doesn't matter which way you go," said the Cat.
"–so long as I get somewhere," Alice added as an explanation.
"Oh, you're sure to do that," said the Cat, "if you only walk long enough."

— Lewis Carroll, Alice's Adventures in Wonderland, 1865

Un científico por lo menos tiene la dignidad de dedicarse a un conocimiento puro. Un ingeniero es un ferretero de la ciencia, que lo único que busca es aplicar los conocimientos que gente superior a él ha encontrado. Es un parásito del conocimiento.

— Adapted from Ignatius Farray (comedian), La Vida Moderna, 2019

Sucking at something is the first step towards being sort of good at something.

— Jake the Dog, Adventure Time

ACKNOWLEDGEMENTS

The submission of my doctoral thesis entails a relief I cannot put into words. It has been a tough and bumpy road, and I have been so focused in arriving to my destination that I have not enjoyed the journey. Fortunately, in this trip I have been accompanied by many people without whose help, support, knowledge and/or care I could not have arrived to where I am now. So thank you to every person that has made this more bearable.

First, I would like to thank my advisor, Alexandre Perera Lluna. Àlex has always believed in me much more than myself, has encouraged me not to give up in my lowest moments and has compensated my constant fight-or-flight state with his easy-going attitude. I am very grateful for his guidance and for helping me to be patient. I also thank Pere Caminal, for his constant interest, support and kindness.

This journey started as a industrial PhD in Mind the Byte, where I learned a lot about start-ups and the world outside university. I want to thank Alfons Nonell the opportunity he gave me, and David Vidal, my supervisor and mentor, for his advice, encouragement and for orienting my academic mind into a more "business" setting. I also deeply thank Melchor, for guiding me the first months, Laura and Héctor, for their help and collaboration, and in general, all my colleagues from Mind the Byte and Intelligent Pharma. Special thanks to Juan for his musical recommendations and for the laughs and complaints sessions in front of a beer.

My journey also passed through the European Bioinformatics Institute in the cold and rainy Cambridge (UK), which was a very rewarding experience both profesionally and personally. I would like to thank my advisor there, Rabie Saidi, for giving me this opportunity, for making me partially recover the enthusiasm for my research and for neutralizing my pessimism with his extreme optimism. I also thank all the people I met there for their warmth and help. And speaking about Cambridge, I am very thankful to Rocío, because having her there made me feel at home.

The highlight of my doctorate years has been the good vibes of the B2SLab/CREB. I am forever grateful for the support from my colleagues, the laughs we shared and the enlighting long lunch-time discussions. They are so nice that I kept going to work even when I stopped getting paid for it. Maria Maqueda, Pol Solà, Jon, Carme, Maria, Joshua, Pol Canal, Enrico, Flavio, Marc Angrill, Ana and all the people that stopped over our office during these years... thank you very much, you deserve the best!

Carrying on with this thesis without despairing would have been impossible without the support of my best friends: Álvaro and Ana. Álvaro, thanks a bunch for listening to me everyday, making me laugh, feeding me and taking care of me. You have followed all this process so closely that I wonder if I should put your name as emotional co-author. Ana, I am so grateful for you being always there for me despite of the distance. You had the words (or cute animal gifs) that I needed in each moment. I also dedicate this thesis to

the rest of AMEBBA (Marta, Elena, Belén and Bea), with whom I grew and whom I owe a very important part of the person I am today; to my UPM friends, my rowing team: Miguel, Celia, Cristina, Mario, Fuen and David, for feeding back our engineering pecculiarities; to my cousin Natalia for her constant support, and to Ana (Galileu), Carme Bauçà and Alicia for making me forget about my thesis with vermouths, concerts, trips and/or walks in the sun.

Many thanks to the amazing open community in Internet (from the tutorials of Machine Learning Mastery to the elaborated responses in Stack Exchange, passing by the amazing Python and R packages developed and maintained by very talented people) from whom I've learned almost everything I know now. Also, thanks to my CrossFit trainer Javi for getting the best out of me physically and boosting my self-confidence, and to La Vida Moderna, which has been there from the beginning of this thesis, making me laugh even in the worst days (Gora Moderdonia Askatuta!).

Finally, I would like to thank my family. To my parents, because their stubborn genetic inheritance is what has brought me to finish this PhD. Thanks for their love, support and for having taught me to be strong and to handle things as they come. To my siblings, Laura and Miguel: Laura, thanks for always being my anchor in life; I've never felt alone nor afraid because I always knew you were there for me. Miguel, thanks for moulding me into the stoical, strategist and pragmatic scientist that I am today. Thanks to Blanca and Alberto for embracing the (not always easy) López del Río idiosyncrasy. And of course, thanks to Mateo and Martín, who did not exist at the beginning of this thesis but now fill my life with cuteness and love.

And last, I warmly thank Sergi because without him this thesis would have never been finished. His patience, encouragement, support, advice, calm and helpful discussions have been the fuel I needed to arrive to my destination. Thank you for having stayed with me during the whole journey. You are the best travel (and quarantine) partner one can wish for.

CONTENTS

LIST OF ABBREVIATIONS

0D	Zero-Dimensional
1D	One-Dimensional
2D	Two-Dimensional
3D	Three-Dimensional
AA	Amino Acid
Acc	Accuracy
Adam	Adaptive Moment Estimation
Adj	Adjusted
ADMET	Absorption, Distribution, Metabolism, Excretion-Toxicity
ANN	Artificial Neural Network
ANOVA	Analysis of Variance
AUC	Area under the (Receiver Operating Characteristic) Curve
AUROC	Area under the Receiver Operating Characteristic Curve
BEDROC	Boltzmann-Enhanced Discrimination of Receiver Operating Characteristic
BMS	Bemis-Murcko Molecular Scaffolds
BRNN	Bidirectional Recurrent Neural Networks
CADD	Computer-Aided/-Aided Drug Design
CI	Confidence Interval
CL	Confidence Limit
CNN	Convolutional Neural Network
cor	Correlation
C-terminal	Carboxyl Terminal
CV	Cross-Validation
CY	Cytochromes P450
df	Degrees of Freedom
DL	Deep Learning
DNA	Deoxyribonucleic Acid
DNN	Deep Neural Network
DUD	Directory of Useful Decoys
DUD-E	Directory of Useful Decoys – Enhanced
ECFP	Extended Connectivity Fingerprint
EC number	Enzyme Commission number
FCFP	Functional Connectivity Fingerprint
FDR	False Discovery Rate
FN	False Negatives
FP	False Positives
fps	Fingerprints
GAN	Generative Adversarial Network
GPCR	G-Protein-Coupled Receptors
GPU	Graphics Processor Units
GR	G-Protein-Coupled Receptors
GRU	Gated Recurrent Unit
IC	Ion Channels

IQR	Interquartile Range
LR	Logistic Regression
LSTM	Long Short Term Memory Networks
Max.	Maximum
MCC	Matthews Correlation Coefficient
MD	Molecular Dynamics
Min.	Minimum
miRNA	Micro Ribonucleic Acid
ML	Machine Learning
mRNA	Messenger Ribonucleic Acid
MUV	Maximum Unbiased Validation
NA	Not Available
NLP	Natural Language Processing
NN	Neural Network
NR	Nuclear Receptors
N-terminal	Amino-Terminal
OE	Other Enzymes
pAUC	Partial Area under the (Receiver Operating Characteristic) Curve
PC	Principal Component
PCA	Principal Component Analysis
PCM	Proteochemometrics
PK	Protein Kinases
Pr	Probability
PR	Proteases
PseAAC	Pseudo-Amino Acid Composition
PseKNC	Pseudo-Nucleotide Composition
PSSM	Position Specific Scoring Matrix
PWM	Position Weight Matrix
Q	Quartile
QMSPR	Quantitative Multiple Structure-Property Relationship
QSAR	Quantitative Structure-Activity Relationship
Qu	Quartile
RF	Random Forests
ReLU	Rectified Linear Activation Unit
RNA	Ribonucleic Acid
RNN	Recurrent Neural Network
ROC	Receiver Operating Characteristic
SD	Standard Deviation
SE	Standard Error
SMOTE	Synthetic Minority Oversampling Technique
Sq	Square
SVM	Support Vector Machine
TN	True Negatives
TP	True Positives
TR	Transporters
TT	Topological Torsions
ZINC	Zinc Is Not Commercial

1 INTRODUCTION

1.1 DRUG DISCOVERY AND DEVELOPMENT

Discovering and bringing a new drug to the market typically takes an average of 14 years of research and clinical development efforts, and costs around 2,000 million euros (DiMasi et al., 2016). Of more than 10,000 hits tested in early drug discovery, only one may eventually lead to a drug that reaches the market [1].

Computer-Aided Drug Design (CADD) methods are used as previous steps to find chemical compounds that bind to the identified target of a certain disease, the so-called hits, eventually becoming a drug candidate. Improving and refining these previous steps in order to select hits more accurately and with more precise information of their properties would imply a small cost that would have huge impact on the time- and money-investment of the whole process.

1.2 PROTEINS

Proteins are macromolecules constituted by amino acid residues. These amino acid residues are covalently attached to one another, forming long linear chains. Proteins can perform a wide range of functions within organisms, including catalysis of metabolic reactions, deoxyribonucleic acid (DNA) synthesis and reparation, response to stimuli, transport of molecules across the cell, reception and sending of chemical signals or structural support. The sequence of amino acids is what primarily identifies and differentiates a protein from the rest. This sequence also defines the specific three-dimensional (3D) structures in which the protein folds, determining its activity, as seen in Figure 1.

Proteins are the most common type of **biological targets**, which means that other entities like endogenous ligands and drugs are directed and/or bind to them. This interaction leads to changes in proteins behavior or function. In pharmaceutical research, the term **target** usually refers to a naturally existing protein involved in some pathology whose activity is modified by a drug, resulting in a specific effect. G-protein-coupled receptors (GPCRs) and enzymes are the classes of proteins that currently predominate drug discovery efforts (Bull and Doig, 2015).

1 https://www.abpi.org.uk/media-centre/blog/2018/august/from-molecule-to-medicine/. Accessed on 16/04/2020

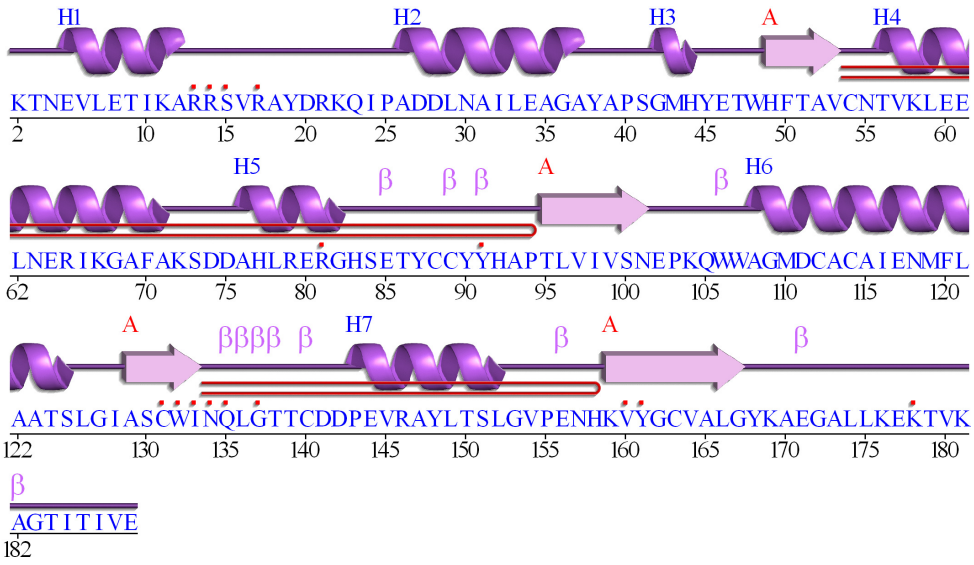

Figure 1: The amino acid sequence of a protein and its corresponding secondary structure topology. Figure obtained from http://www.topsan.org/, TOPSAN database for structural genomics (Ellrott et al., 2011), accessed on 08/08/2017.

1.3 SMALL MOLECULES

Most of the currently used drugs are small molecules that interact with proteins. **Small molecules** are low molecular weight organic compounds (<900 daltons) that may help regulate biological processes. This size limit allow molecules to easily diffuse across cytoplasmic membranes and reach intracellular sites of action (Veber et al., 2002). In addition, this cutoff is a necessary condition for oral bioavailability (Veber et al., 2002). As seen before, in pharmacological research this term would be restricted to a molecule (**ligand**) that binds to a specific biological target (see Figure 2), altering its activity or function. These molecules can be natural (such as secondary metabolites) or artificial.

If a molecule binds to a certain target it is called *active*, and *inactive* if it does not. Since a limited number of inactive molecules has been published in literature, compounds structurally similar to actives but experimentally not tested for biological activity are used as putative inactive molecules, also called **decoys**.

1.4 DEEP LEARNING BASICS

Deep learning (DL) is a branch of machine learning (ML) whose computational models to make data-driven predictions are composed of multiple non-linear transformations (processing hidden layers) to yield abstract representations of data (LeCun et al., 2015). Unlike ML methods, DL is able to be fed with raw data, without the need of a domain expertise to *a priori* engineer a set of representative features. Moreover, well-regularized deep neural networks (DNNs) can exploit commonalities between different tasks to transfer knowledge.

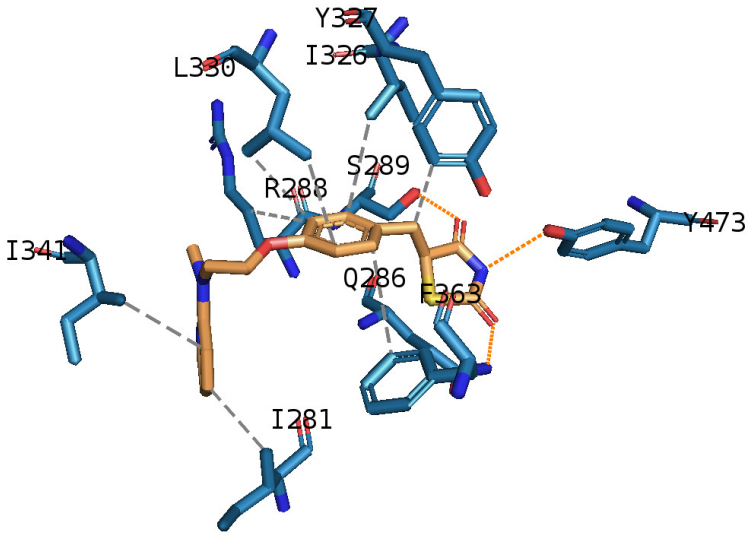

Figure 2: Example of interaction diagram generated with PLIP software (Salentin et al., 2015). A protein (blue) binding a molecule (ochre). Dotted lines represent the different interactions between them. Alphanumeric codes represent the type and position of binding amino acid residues.

DL methods have drastically improved the state-of-the-art in multiple domains, including natural language processing (NLP) (Vaswani et al., 2017), image (K. He et al., 2016) and speech recognition (Hannun et al., 2014) or computational biology (Jones et al., 2017). Specifically in the computational biology field, DL seems well suited since its mechanism is able to deal with complex patterns in nature (Gawehn et al., 2016). DL has also proven to be succesful for drug development (H. Chen et al., 2018; Dahl et al., 2014; Lusci et al., 2013; Ramsundar et al., 2015; K. Tian et al., 2016), but this topic will be further developed in next chapter.

Most DL algorithms are based on artificial neural networks (ANNs), an algorithm modeled after biological neural networks used to estimate functions by translating a large number of inputs into a target output. ANNs are composed of layers, each one comprising many nodes (*neurons*) (see Figure 4). Each of these neurons has its own parameters, called *weights* $\boldsymbol{w}$ and *bias* b. Every neuron accepts input values $\boldsymbol{x}$ from the previous layer, multiplies them with its weights vector and adds its bias. The resulting value is then mapped into a nonlinear function $g(z)$ (see Figure 3). The output of a single neuron z would be the following:

$$\hat{y} = g(\boldsymbol{w}^{\boldsymbol{T}} \cdot \boldsymbol{x} + b) \tag{1}$$

$\hat{y}$ is then an input for the next layer in the ANN and so forth. The output of the last layer corresponds to the objective to be predicted. Moreover, the weights and bias of each neuron's function is adjusted in the construction of the model to minimize the error of the predicted value (Goh et al., 2017b).

For the construction of a DNN (i.e., ANN with multiple hidden layers, see Figure 4) it is necessary to determine how to assign error attribution and make corrections to weights by going backwards from the predicted

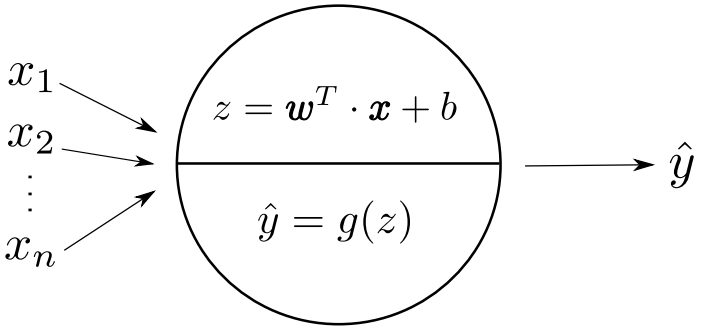

Figure 3: Schematic representation of a neuron. Each neuron has it weights $\boldsymbol{w}$ vector and its bias b. These are multiplied and added, respectively, to the inputs from the previous layer $\boldsymbol{x}$. The result of passing it to a nonlinear activation function $g(z)$ is the output of the neuron.

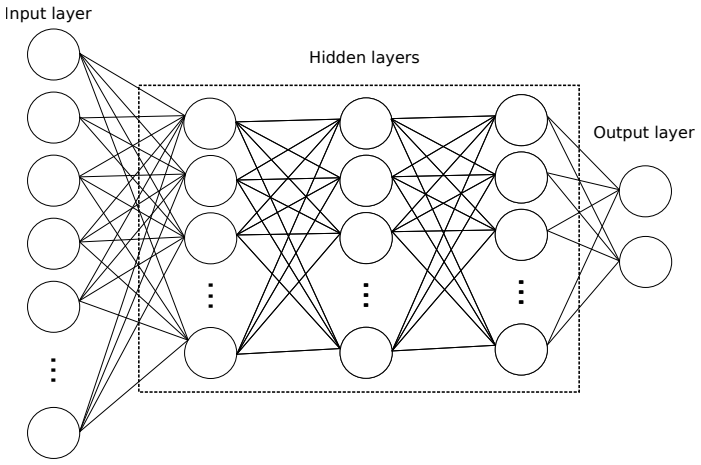

Figure 4: Schematic representation of a feed-forward DNN, also called fully connected. Each neuron, denoted as circles, accept a series of n input values and maps it to an output using a nonlinear function.

output through the neural networks, which is known as **backpropagation** (Rumelhart et al., 1986). During this process, an algorithm called **stochastic gradient descent** is used to find the minimum in the error surface caused by each neuron when generating a corresponding output. An error function of the target output of the DNN is iteratively minimized and the weights of the neurons are updated each iteration. The data in the training set may be iterated over multiple times, being an *epoch* a complete pass over the data (Goh et al., 2017b).

A key issue with backpropagation is the *vanishing gradient problem*: the error signals become progressively more diffused as the signal goes back through each hidden layer. In addition, ANNs are very susceptible to undergo overfitting, *i.e.* learning particular cases without being able to generalize for new data. To improve the training process for ANNs, several important regularization techniques have been developed, as will be further explained in section 2.2. Some examples are *rectified linear activation* (ReLU) function (Glorot et al., 2011), which solves the vanishing gradient problem because of its first derivative being 1 or 0; or the *dropout* algorithm (N. Srivastava et al., 2014), in which for each epoch of the training process, a fixed proportion of neurons are randomly selected to be temporarily excluded from the model, reducing overfitting.

In addition to this traditional feedforward DNN, more recent developments include alternative and more specific architectures as convolutional neural networks (CNN) or recurrent neural networks (RNN), which will be further explained in next chapter.

1.5 DESCRIPTORS

In order to train ML-based models, it is necessary to obtain or generate features that identify and characterize molecules and proteins. **Descriptors** are the transformation of the symbolic representation of chemical information into numbers, allowing some mathematical treatment of molecules.

1.5.1 Molecular descriptors

Currently it is possible to generate more than 3000 different types of molecular descriptors (Todeschini and Consonni, 2008) describing structural and physico-chemical features of the ligand. The information contained in a descriptor depends on the molecular representation of the compound and the algorithm used for the computations. Molecular descriptors are distinguished by their data type (*e.g* boolean, integers or real numbers) and the dimensionality of the molecular representation from which they are calculated (Andersson et al., 2011). According to their dimensionality, they can be zero-dimensional (0D), derived from the chemical formula; one-dimensional (1D), derived from the molecule representation as a substructure list; two-dimensional (2D), calculated from the graphical representation of the chemical structure and 3D descriptors, computed from the 3D conformations (which is unknown for most ligands) (Andersson et al., 2011). The higher

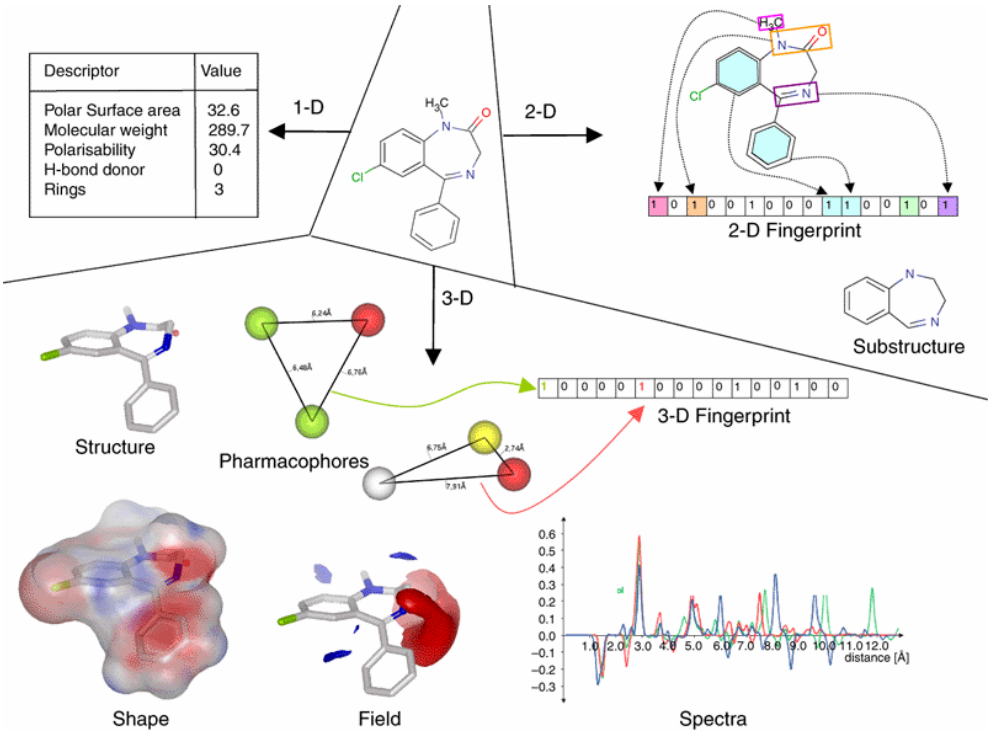

Figure 5: Examples of the different types of molecular descriptors. Figure reproduced from (Rognan, 2007) with permission of John Wiley and Sons (17/04/2020).

the dimensionality, the better they establish relationship between structure and function but the more difficult they are to interpret. A summary of the main types of descriptors can be seen in Figure 5.

1.5.2 Protein descriptors

For protein representation it is also possible to generate more than 1000 features from its amino acid sequence (Cao et al., 2013; Z.-R. Li et al., 2006). Protein descriptors are divided in two types: alignment-dependent and alignment-independent. *Alignment-dependent descriptors* require a dataset of alignable proteins, so this kind of descriptors are only useful if proteins are similar in sequence (Andersson et al., 2011). *Alignment-independent descriptors* can be computed from the amino acid sequence, covering amino acid composition, distribution and transition (Andersson et al., 2011). These are easy to interpret and are mainly used to classify proteins by function or modeling protein-ligand interaction. However, descriptors computed from the primary structure of the protein do not take into account binding site location since amino acid sequence has no structural information about the protein.

2 STATE OF THE ART

2.1 CHEMOGENOMICS

Chemogenomics is an emerging approach to drug discovery that lies in the interface of biology, chemistry and informatics (Bredel and Jacoby, 2004). Its goal is to measure the response of all gene products encoded by the human genome (i.e. all proteins) to all available chemical compounds.

However, the size of the protein and ligand space make impossible a systematic *in vitro* approach to this aim: in humans, the estimated number of proteins is more than one million (O'Donovan et al., 2001), while the number of reasonably sized molecules that contain atoms commonly found in drugs rises up to 10^{62} (Lipinski, 2000). Because of this, interaction data can only be generated for a minute fraction of all possible protein-ligand complexes.

To overcome this bottleneck and accelerate the drug discovery process, large collections of chemical products are **virtually screened** for the identification both of biological targets and biologically active compounds. Chemogenomics-based drug discovery approaches are traditionally classified in three categories: methods focused on *ligands*, *targets* or *the combination between ligands and targets*. However, in many studies there is a combination of different approaches.

2.1.1 Ligand-based methods

This group of methods are based on the paradigm that molecules that are similar enough to existing annotated ligands are expected to have similar biological activity (Rognan, 2007). Molecular descriptors from compounds known to be active for a certain protein are computed (see Table 2). A set of mutual characteristics of a compounds series is identified and used as *molec-*

Table 2: The similarity of ligands and targets can be defined using different methods or descriptors. Figure adapted from (Klabunde, 2007) with permission of John Wiley and Sons (20/04/2020).

Ligand similarity	Target similarity
Chemical scaffold	Target class (e.g. GPCR)
Chemical fragment	Target subclass (e.g. purinergic GPCR)
Chemotype (2D fingerprints)	Overall sequence homology (phylogenetic tree)
3D pharmacophore	Similarity of binding site (3D structure or 1D sequence motifs)

ular filters, which are used to select compound for experimental evaluation, reducing the chemical space (Ferreira et al., 2015).

The most used ligand-based approach is **Quantitative Structure-Activity Relationship** (QSAR) (Sawada et al., 2014). In QSAR, one series of compounds is characterized by various descriptors. Each ligand is linked to an experimentally measured biological activity against a certain target and a model is induced by ML methods. This model is used for prediction of binding affinity of new ligands for said target. Additionally, the analysis of the model may provide information on which are the most relevant features for activity prediction for the studied target.

Predictive QSAR models have been successfully used as the first step of CADD in studies, some of them even being subsequently experimentally validated and patented. Some examples include the discovery of anticonvulsants (Brown et al., 2014; M. Shen et al., 2004), HIV-1 reverse transcriptase inhibitors (Medina-Franco et al., 2005), dopamine D1 antagonists (Oloff et al., 2005), antitumour compounds (S. Zhang et al., 2007), beta-lactamase inhibitors (Hsieh et al., 2008), Trk receptor antagonists (Tammiku-Taul et al., 2016) or human histone deacetylase inhibitors (H. Tang et al., 2009). In a review of the QSAR-based therapeutic patents from 2000 to 2018 (Halder et al., 2018) it is noted the growing presence of novel compounds for neurodegenerative diseases discovered by these techniques, *e.g.* yohimbine-derivated anti-psychotics (S. K. Srivastava et al., 2012) or amyloid binding alcohol dehydrogenase inhibitors for Alzheimer's disease (Yan and Valasani, 2017).

Another ligand-based approach is the extraction of structural features from known ligands to generate 3D **pharmacophore** models, which are abstract representations of the molecular properties that are necessary for the molecular recognition from a certain target (Ferreira et al., 2015; Kutlushina et al., 2018). This 3D models take into account important information that 2D descriptors do not seize related to the binding site and kind (whether the molecule acts as an agonist or an antagonist). However, this approach is limited by the sparsity of the crystallographic data (Rognan, 2007) and by the restrictions of the available current free ligand-based pharmacophore modeling tools (Kutlushina et al., 2018).

Nevertheless, these ligand-based approaches do not work well when there are few ligands known for a protein of interest. Moreover, ligands usually bind to more than one target, which can be overlooked if centering the analysis only to the interaction with one certain protein. This could cause unwanted side effects from cross-interactions between a drug and other proteins in the proteome.

2.1.2 Target-based methods

Target-based methods aim to select ligands by comparison and classification of related targets based on ligand binding sites by using sequence motifs or 3D structural information (see Table 2). This approach mainly focuses on those residues known to be important for the binding.

Sequence-based approaches are centered in multiple alignments, and they are specially useful for protein families with a lack of high-resolution struc-

tural data, such as GPCRs (Granier and Kobilka, 2012). It has been found that residues in the binding sites of GPCRs can be extracted an concatenated into a short ungapped sequence which can be later used to derive a distance matrix based on sequence identity (Surgand et al., 2005) or physicochemical properties (Frimurer et al., 2005). This ungapped sequence reproduces perfectly the full sequence-based phylogenetic tree, suggesting that only a few residues are relevant when comparing targets across a family.

An example of the application of these cavity-based trees is in Frimurer et al., 2005, where the ligand-binding cavity of two receptors was found to be closely resemblant with respect to the physico-chemical properties of the residues forming the binding sites although both targets shared only a low overall sequence homology. Several hits found by virtual screening for the second receptor passed successfully experimental tests on the first receptor.

Structure-based methods compare structural templates of ligand-binding sites of the targets. They require a protein 3D structure of good quality, which is limited only to certain protein families. Moreover, this approach is highly dependent on the structural alignment and the grid resolution, and only targets of the same family can be compared. However, it has been successfully applied to different protein families such as protein kinases, serine proteases and nuclear hormone receptors (Hoppe et al., 2006; Naumann and Matter, 2002; Pérot et al., 2010).

2.1.3 Protein-ligand interaction-based methods

Unlike the previous approaches, these methods attempt to predict ligands for a target of interest taking into account information from both the ligand and the target. The two dominating directions are classical *physics-based* docking methods and ML-based *proteochemometrics*.

Prediction of protein-ligand interaction is used to screen large drug-like compound libraries in order to get leads that will serve as input for subsequent phases in the CADD pipeline.

Classical physics-based techniques

The classical approach of calculating binding affinity of a small molecule and a protein is using docking simulations (Ferreira et al., 2015) based on physics. **Molecular docking** (see Figure 6) is a family of optimization-based computational algorithms that attempt to predict the preferred orientation and the binding affinity of one molecule to a second structure, usually a protein. In order to make that simulation, experimental holo-structures of the protein are needed, which leads to the main disadvantage of docking simulations: it cannot be used for proteins whose 3D structure is unknown (Sawada et al., 2014).

There is a variety of docking algorithms and servers available, such as GOLD (Verdonk et al., 2003), Glide (Friesner et al., 2004), DOCK Blaster (Irwin et al., 2009), SwissDock (Bitencourt-Ferreira and de Azevedo, 2019; Grosdidier et al., 2011), rDock (Ruiz-Carmona et al., 2014), istar (H. Li et al., 2014), DOCK6 (Allen et al., 2015) or the AutoDock suite (Forli et al., 2016). Docking techniques have been also recently questioned (Y.-C. Chen

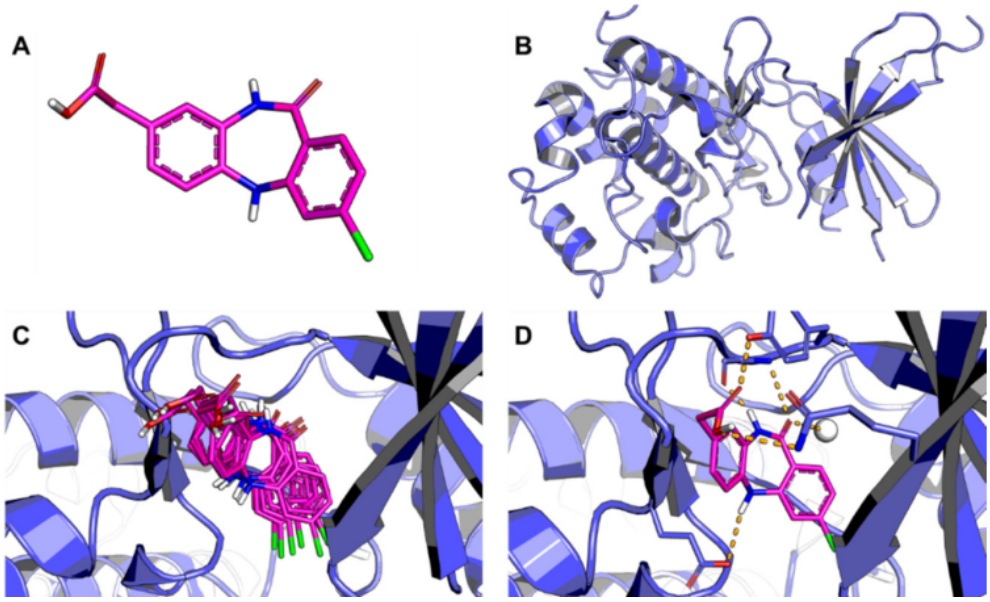

Figure 6: Outline of the molecular docking process. **A.** 3D structure of the ligand; **B.** 3D structure of the receptor; **C.** The ligand is docking into the binding cavity of the receptor and putative conformations are explored; **D.** The most likely binding conformation and the corresponding intermolecular interactions are identified. Figure obtained from Ferreira et al., 2015 (permission granted by the open access Creative Common CC BY license).

and Apostolakis, 2015). In Plewczynski et al., 2011 no correlations could be observed between the most popular docking programs score and *in vitro* binding affinities.

In the traditional workflow, once the optimal conformation of the molecule based on the binding energy is obtained from docking simulations based on binding energies, **molecular dynamics** (MD) simulations are performed (Alonso et al., 2006). MD takes into account flexibility and motion of the target binding sites, which in some cases undertake significant conformational changes during molecular recognition. MD is used to estimate the stability of a ligand-receptor complex proposed by molecular docking (Salsbury and Jr., 2010). Although not as wide as in the case of molecular docking, there is also available a collection of algorithms to perform MD simulations, such AMBER (Cornell et al., 1996; Salomon-Ferrer et al., 2013), CHARMM (B. R. Brooks et al., 2009), GROMOS (Christen et al., 2005) and GROMACS (Abraham et al., 2015).

Despite of its contributions to molecular docking, MD has some important limitations, noting the high computation cost demanded by the simulation of large systems and the fact that some of the conformational changes of the target occur on time scales that exceed computational capacity (Ferreira et al., 2015). Additionally, available force fields for MD studies have some deficiencies, like bias towards particular amino acid residues or secondary structures, misrepresentation of crowded environments of the cell or the limitation on parameters for post-translational modifications (Kumari et al., 2017).

Proteochemometrics

The aim of proteochemometrics, also called **Quantitative Multiple Structure-Property Relationship** (QMSPR), is to model the interaction of both target and ligand and predict a property of the interaction, usually the binding affinity value (Andersson et al., 2011). For this, a data matrix is built, each of its rows containing descriptors of a protein-ligand complex linked to some experimentally measured biological activity. A statistical or ML method is used to induce the model. The prevalent ML architectures used to this end include linear and non-linear principal component analysis (PCA), k-means clustering algorithms, partial least square projection to latent structures, decision trees, multivariate linear regression, linear discriminant analysis, support vector machines (SVM), logistic and kernel regression, multi-layer perceptrons and related neural networks approaches (Gawehn et al., 2016).

The induced model can be applied for predictions of interaction with new proteins as well as ligands. Interpretation of the model may provide information on which ligand or target properties are important for prediction of biological activity. Finally, the model is validated, either *in silico* (cross-validation, external test set, MD simulations) or *in vitro* by binding affinity assays. The proteochemometrics modeling process is illustrated in the workflow diagram in Figure 7.

Proteochemometrics is a conceptual extension of QSAR (see Figure 8), where the target protein was fixed and biological activity was predicted only from ligand descriptors. Each row thus contains an unique combination of protein and ligand descriptors linked to some biological activity, allowing to build models that cover several series of ligands and targets (Qiu et al., 2017).

Main advantages of this approach are that interaction data stored in many different databases and in the literature can be reused; models can predict cross-interactions of a given ligand to other proteins in the proteome; and different binding and interaction features missed in conventional QSAR studies can be used for the predictions.

According to Qiu et al., 2017, proteochemometrics modelling has multiple applications thanks to its ability to study multiple target-ligand interactions. It is widely used to study the mechanism of molecular recognition at different levels of interaction. It is also applied for target inhibitor screening in pathogenic viruses and major diseases, such as cancer. Proteochemometrics is also a common tool in other scopes such as peptide-protein interaction, specific protein-protein interaction and antigen-antibody interaction .

Specifically, the QMSPR approach has been applied to different drug targets families such as melanocortin GPCRs (Lapinsh et al., 2003), adenosine receptors (Kramer and Gedeck, 2011), HIV proteases (Q. Huang et al., 2012; Lapins et al., 2008; Lapins and Wikberg, 2009), histone deacetylases (D. Wu et al., 2012), cytochtome P450 enzymes (Kontijevskis et al., 2008) or the target proteins of *Plasmodium Falciparum* and *Toxoplasma gondii* (Paricharak et al., 2015), among others. Also, large-scale proteochemometrics studies have been performed screening thoroughly public databases (Dakshanamurthy et al., 2012; Sawada et al., 2014; Strömbergsson et al., 2010).

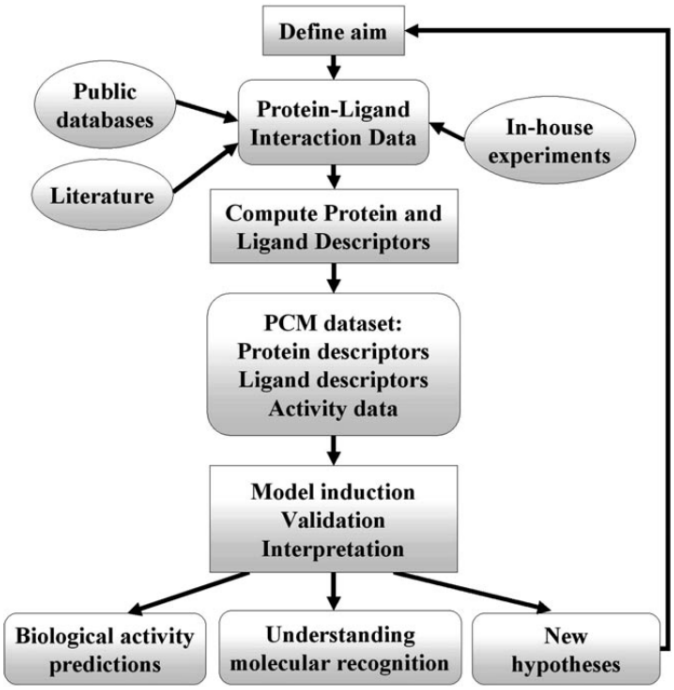

Figure 7: The main steps in proteochemometrics modelling process. PCM: Proteochemometrics. Figure reproduced from Strömbergsson et al., 2010 with permission of John Wiley and Sons (22/04/2020).

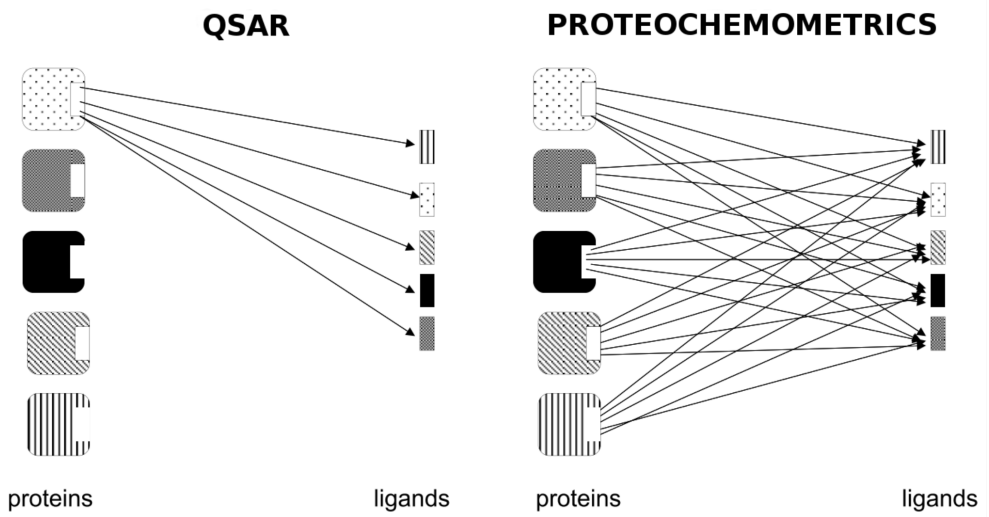

Figure 8: Schematic comparison between QSAR and proteochemometrics. A QSAR model is based on one series of ligands interacting with one target, whereas in proteochemometrics several series of ligands and proteins are combined to build a single model. Figure reproduced from Andersson et al., 2011 with permission of Bentham Science Publishers LTD (22/04/2020).

2.2 ADVANCEMENTS IN DEEP LEARNING

In the introduction, the foundations of ANNs were introduced. These concepts started to be developed in the 1940s (McCulloch and Pitts, 1943) and made some background progress throughout the years (LeCun et al., 1998; Rosenblatt, 1958; Rumelhart et al., 1986) but without being really adopted by the scientific community, especially after the SVMs became the method of choice (Cornell et al., 1996). It was not until 2012 that there was a breakthrough of the ANNs, rebranded as DL networks (G. E. Hinton et al., 2006): in the ImageNet competition, deep convolutional networks were applied to more than a million labeled images, almost halving the error rates of the best competing approaches (Krizhevsky et al., 2012). From this moment, DL has experienced a resurgence to which different factors have contributed:

- Availability of **large labeled datasets** coupled to the outbreak of big data (see section 2.3).
- Increase of computational power by parallel computing with **Graphics Processor Units** (GPUs), traditionally associated to the computer-game industry. GPUs pack thousands of relatively simple processing cores on a single chip, which is suitable for the neural networks computations.
- Resolution of the vanishing gradient problem by the application of the previously introduced ReLUs and **non-saturating activation functions** in general.
- **Improved architectures**, which will be deepened in this section.
- **Software platforms** like Tensorflow (Martín Abadi et al., 2015), Theano (Theano Development Team, 2016), PyTorch (Paszke et al., 2019) and wrappers that run on top of these frameworks like Keras (Chollet et al., 2015) or Lasagne (Dieleman et al., 2015), that allow to make proptotyping faster, less error-prone and automatically managing GPU computing.
- New **regularization** techniques that reduce overfitting, like the already mentioned *dropout*, *batch normalization* (Ioffe and Szegedy, 2015), which is a method to reduce internal covariate shift in neural networks or *data augmentation* techniques to create more training examples by deforming the existing ones.
- Robust **optimizers**: modifications of the stochastic gradient descent algorithm introduced in the introduction, including the *Adaptive Gradient algorithm* (AdaGrad) (Duchi et al., 2010), *Root Mean Square Propagation* (RMSprop) (G. Hinton et al., 2012) and *Adaptive Moment Estimation* (Adam) (Diederik P Kingma and Ba, 2014).

However, it must be stated that many DL improvements have been driven by empirical performance on a standard set of benchmarks, having a limited theoretical justification.

In this section, more complex, specific and problem-oriented architectures and algorithms will be presented. Different combinations of these advancements constitute the state-of-the-art in DL applications.

2.2.1 Convolutional neural networks

CNNs are designed to process data in the form of multidimensional arrays. Although CNNs are usually associated to 2D image processing (Krizhevsky et al., 2012; LeCun et al., 1989), other data modalities come in this form, such as 1D sequences (Alipanahi et al., 2015; Angermueller et al., 2016). The high dimensionality of these data would result in a excessive number of parameters for a fully connected neural network model. To avoid this, CNNs make assumptions on the structure of the network, reducing so the effective number of parameters to learn.

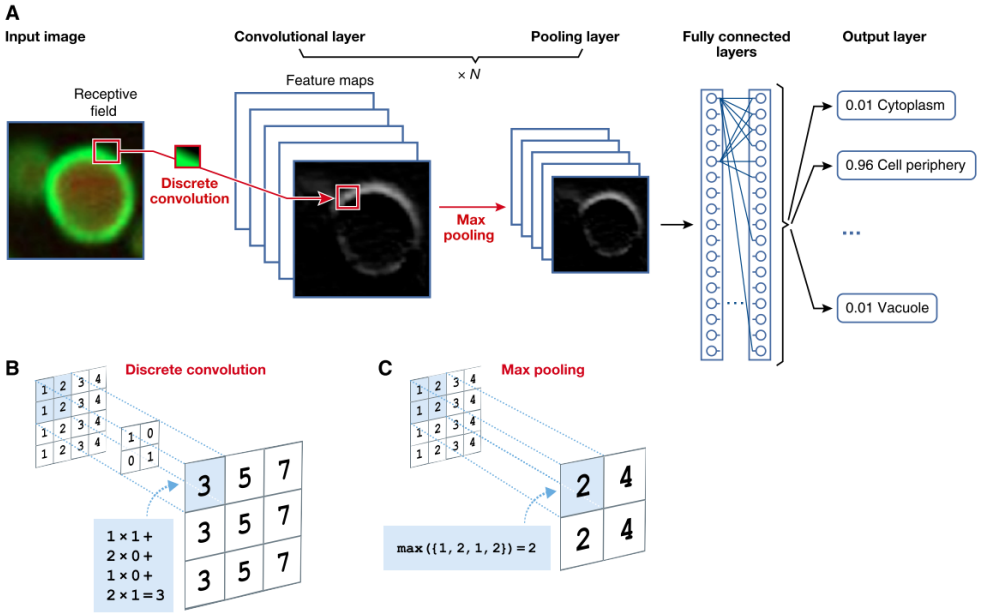

Figure 9: **A.** Schema of a CNN applied to the image of a cell. **B.** Discrete convolution performed in feature maps. **C.** Summarization carried out by the pooling layer. Figure obtained from Angermueller et al., 2016 (permission granted by the open access Creative Common CC BY license).

Following the notation of Angermueller et al., 2016, a convolutional layer is formed by multiple **feature maps** of the same size of the input associated to **filters** of a given size (see Figure 9 A). Each neuron within a feature map is only connected to a local patch of neurons in the previous layer, the *receptive field* (local connectivity). All neurons within a feature map share the same parameters (*parameter sharing*). Thus, all neurons within a feature map scan for the same feature in the previous layer at different locations. Different feature maps could, for instance, detect edges of different orientations in an image or certain patterns or motifs in a sequence. The activity of a neuron is obtained by computing a discrete convolution of its receptive field and applying an activation function (Figure 9 B).

If the input is represented as X and a filter as f, the mathematical expression of the operations performed to compute the activity of the neurons in the feature map associated to f is

$$Z = X * f. \tag{2}$$

Convolutional layers imply translational invariance (Kauderer-Abrams, 2017). Often the exact position and frequency of features is not important for the final prediction. Under this assumption, the **pooling** layer summarizes adjacent neurons by computing, for example, the maximum or the average over their activity (Figure 9 C). By applying the same pooling to small patches shifted by more than one pixel, the input is effectively down-sampled, reducing the number of model parameters (Angermueller et al., 2016).

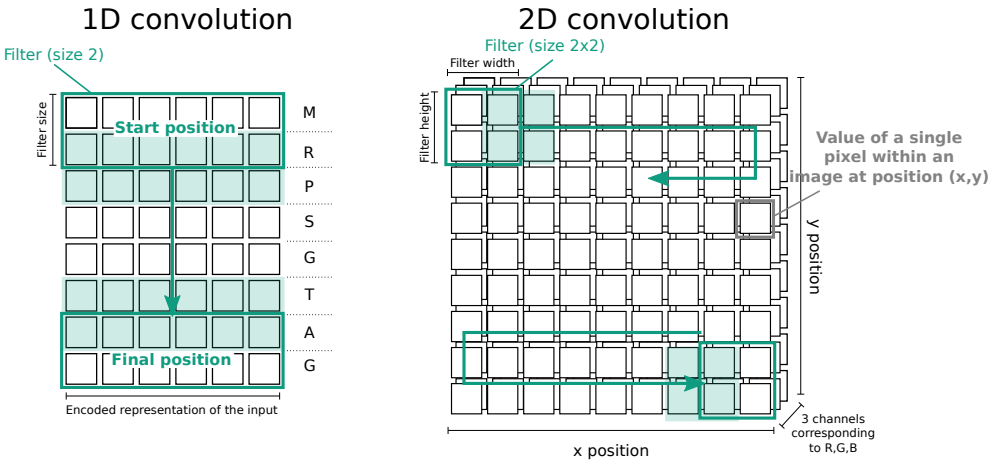

Figure 10: Comparison between 1D and 2D convolutional layer. Figure inspired in "1D versus 2D CNN" by Nils Ackermann[1], permission granted by Creative Commons CC BY-ND 4.0.

1D convolutional layers follow the same approach as 2D convolutional layers. The main differences are the dimensionality of the input data and how the filter slides across it. A comparison is shown in Figure 10. In the left there is an example of a 1D convolution with a biological sequence composed of 8 elements as input. Each element is encoded in a low dimensional representation. The filter always covers the whole codification of an element. The size of the filter determines how many elements are considered when training it (in this example, 2). In the right, there is an example of a 2D convolution on an image. Each pixel is represented by its x and y position and three values (R, G, B). The filter has a dimension of 2x2 in the example and will slide both horizontally and vertically across the image.

A typical CNN consists of a combination of multiple convolutional and pooling layers, to learn more abstract representations of data, followed by one or more fully connected layers.

1 https://blog.goodaudience.com/introduction-to-1d-convolutional-neural-networks-in-keras-for-time-sequences-3a7ff801a2cf. Accessed on 28/04/20.

2.2.2 Recurrent neural networks

Standard neural networks rely on the assumptions of independence among the training and test examples and inputs of the same length. After each data point is processed, the entire state of the network is lost (Lipton et al., 2015). For modeling data with temporal or sequential structure, varying length inputs and outputs and long-range dependencies, RNNs are better suited (Hopfield, 1982).

RNNs are connectionist models that selectively pass information across sequence steps, while processing sequential data one element at a time (Lipton et al., 2015), as seen in Figure 11. This allows information to persist. Mathematically, for each timestep t, the activation A_t of the neural network (sometimes also called *hidden node*) and the output h_t are expressed as follows [2]:

$$\boldsymbol{A_t} = g_1(w_{AA}\boldsymbol{A_{t-1}} + w_{Ax}\boldsymbol{x_t} + \boldsymbol{b_A}) \tag{3}$$

$$\boldsymbol{h_t} = g_2(w_{hA}\boldsymbol{A_t} + \boldsymbol{b_h}) \tag{4}$$

where w_{Ax}, w_{AA}, w_{hA}, $\boldsymbol{b_A}$, $\boldsymbol{b_h}$ are cofficients that are shared temporally and g_1, g_2 activation functions. Specifically, according to the notation of Lipton et al., 2015, w_{Ax} is the matrix of conventional weights between the input and the activation of the NN and w_{AA} is the matrix of recurrent weights between the activation of the NN and itself at contiguous time steps. The vectors $\boldsymbol{b_A}$ and $\boldsymbol{b_h}$ are the bias parameters. These equations mean that at time t, nodes from RNNs receive input from the current data point x_t and also from the activation values from the network's previous state A_{t-1} . The output h_t at each time t is calculated given the activation values A_t at time t. Thus, input x_{t-1} at time $t-1$ can influence the output h_t and later through recurrent connections (Lipton et al., 2015).

RNNs have been successfully used to model both sequential inputs and sequential outputs as well as mappings between single data points and sequences (in both directions). Their main advantages are being able to process inputs of any length and weights sharing across time. RNN models are mostly used in the fields of NLP (K. Cho et al., 2014) and speech recognition (Miao et al., 2015).

However, learning with RNNs has long considered to be difficult due to their computational complexity and their problem handling long-range dependencies (Bengio et al., 1994; Hochreiter et al., 2001). The problems of *vanishing* and *exploding* gradients occur when backpropagating errors across many time steps: if the weight along the recurrent network is less than one, the contribution of the input at the first time step will decrease exponentially fast as a function of the length of the interval in between, and *vice versa*.

Truncated backpropagation through time is a partial solution to the exploding gradient problem for continuously running networks (Williams and Zipser, 1989): a maximum number of time steps is set along which error can

2 https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-recurrent-neural-networks#word-representation. Accessed on 28/04/2020

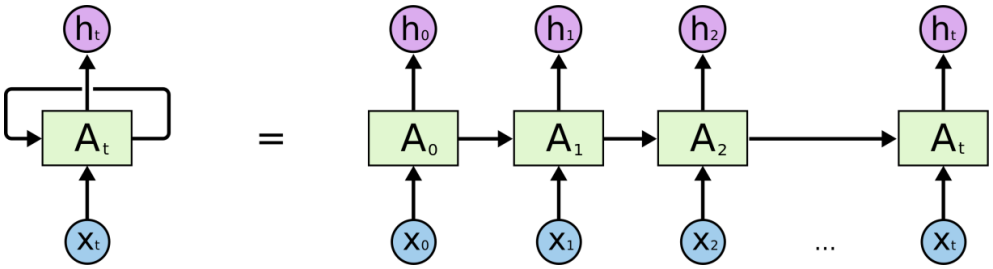

Figure 11: In the left, a RNN at timestep t. A neural network (its activation referred as A_t) looks at some data point x_t and outputs a value h_t. A loop allows information to be passed from one step of the network to the next. In the right, this process unrolled: Multiple copies of the same network, each passing a message to a sucessor. Figure adapted from `http://colah.github.io/posts/2015-08-Understanding-LSTMs/`, permission granted by the author (23/04/20).

be propagated. However, it implies the loss of its ability to learn long-term dependencies, which is needed in most applications.

Long short-term memory networks

Long Short Term Memory networks (LSTMs) is a special architecture of RNN capable of learning long-range dependencies (Hochreiter and Schmidhuber, 1997). They introduce the concept of the **memory cell**, a unit of computation that replaces traditional nodes in the hidden layer of a network. Each memory cell contains a node with a self-connected recurrent edge of fixed weight one, ensuring that the gradient can pass across many time steps without vanishing or exploding (Lipton et al., 2015).

According to Lipton et al., 2015, simple RNNs have *long-term memory* in the form of weights that change slowly during training, storing general knowledge about the data. They also have *short-term memory*: ephemeral activations passing from one node to its contigous. LSTM models introduce an intermediate type of storage via the memory cell. A memory cell is composed of simple nodes in a specific connectivity pattern, with novel multiplicative nodes (represented by Π).

Since the original LSTM was introduced, several variations have been proposed. *Forget gates* (Gers et al., 2000) provide a method by which the network can learn to flush the contents of the internal state, which is especially useful in continuous running networks. Due to their proven effectiveness, nowadays they are standard in most modern implementations (Lipton et al., 2015).

A schema of a LSTM memory cell with a forget gate can be seen in Figure 12. Following the notation on Lipton et al., 2015, all elements are enumerated and described below (the subscript c is used to refer to an individual memory cell):

- *Input node* g_c: this node takes the activation from the input layer $\boldsymbol{x_t}$ at the current time step t in the standard way and from the hidden layer in the previous timestep A_{t_1}. The summed weighted input is then passed through a nonlinear activation function.

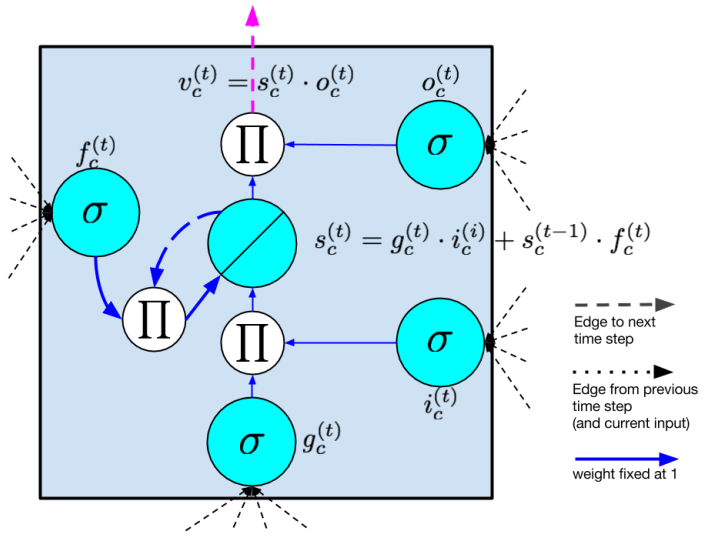

Figure 12: LSTM memory cell as proposed in (Hochreiter and Schmidhuber, 1997) with a forget gate (Gers et al., 2000). The self-connected node is the internal state (the diagonal line indicates that it is linear). The blue dashed line is the recurrent edge, which has fixed unit weight. Nodes marked Π output the product of their inputs. All edges into and from Π nodes also have fixed unit weight. Figure obtained from Lipton et al., 2015 (permission granted by the open access Creative Common CC BY license).

- *Input gate* i_c: a gate is a sigmoidal unit that takes activations from the current data point x_t and from the hidden layer at the previous time step, like the input node. The value of a gate is used to multiply the value of another node, so if its value is 0, then flow from the other node is cut off. If the value is 1, all flow is passed through. Thus, the value of the the *input gate* multiplies the value of the *input node*.

- *Internal state* s_c: it is a node with linear activation. It has a self-connected recurrent edge with fixed unit weight. Since this edge spans contigous time steps with a constant weight, error can flow across time steps without vanishing or exploding. The update of the internal state is

$$\boldsymbol{s_t} = \boldsymbol{g_t} \odot \boldsymbol{i_t} + \boldsymbol{s_{t_1}}. \tag{5}$$

- *Forget gate* f_c: it has been already mentioned that they give a method for the network to flush the internal state. With forget gates, the equation to calculate the internal state of the forward pass is

$$\boldsymbol{s_t} = \boldsymbol{g_t} \odot \boldsymbol{i_t} + \boldsymbol{f_t} \odot \boldsymbol{s_{t_1}}. \tag{6}$$

- *Output gate* o_c: the final value v_c produced by a memory cell is the value of the *internal state* s_c multiplied by the value of the *output gate* o_c.

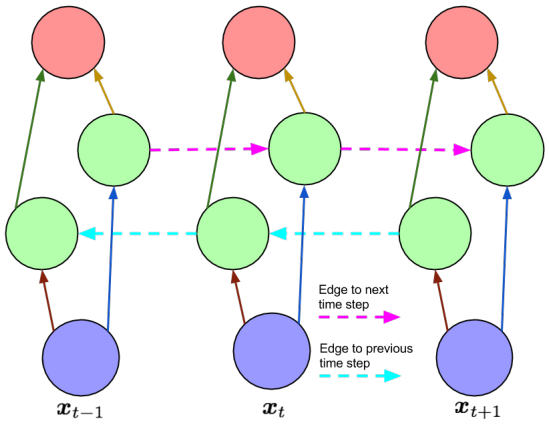

Figure 13: A BRNN as described in Schuster and Paliwal, 1997 unfolded in time. Figure obtained from Lipton et al., 2015 (permission granted by the open access Creative Common CC BY license).

Other types of RNNs

BIDIRECTIONAL RECURRENT NEURAL NETWORKS

In traditional RNNs, only past input can affect the output. Bidirectional recurrent neural networks (BRNN) are an architecture in which information from both the future and the past is used to determine the output at any point in the sequence, solving this issue (Schuster and Paliwal, 1997).

In this architecture (Figure 13) there are two layers of hidden nodes. Both hidden layers are connected to input and output: the first has recurrent connections from the previous steps, while in the second the direction of recurrent connections is flipped, passing activation backwards along the sequence.

The limitations of BRNN is that it is only appropiate for prediction over a sequence of *fixed* length, and never on a online setting. It must also be said that LSTM and BRNN are not mutually exclusive and have been successfully combined (Graves et al., 2012; Graves and Schmidhuber, 2005).

GATED RECURRENT UNITS

Gated recurrent units (K. Cho et al., 2014) (GRUs) are another approach for solving the vanishing gradient problems of the classical RNNs. Compared with LSTMs, GRUs lack of *internal state* and use the hidden state to transfer information. Moreover, it only has two gates: a *reset gate* and an *update gate* [3]. The *update gate* combines the forget and input gates of the LSTMs, deciding what information to keep and what information to flush. The *reset gate* is also used for deciding how much information to forget. The resulting model is more efficient computationally than standard LSTM models and

3 https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21

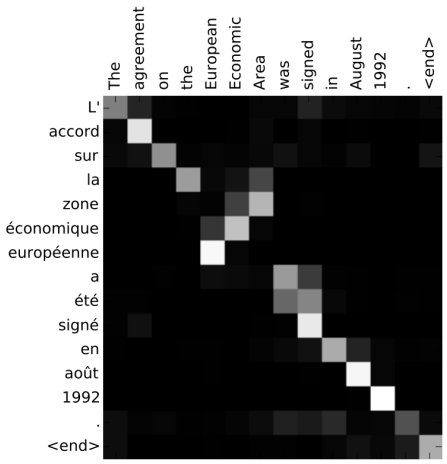

Figure 14: Alignment of words in an English-French translation (Bahdanau et al., 2014). From the annotation weights represented, it can be seen which positions in the source sentence were considered more important when generating the target word (permission granted by the open access Creative Common CC BY license).

obtains similar performance (Chung et al., 2014). This is the reason why GRUs have been growing increasingly popular in the last years [4].

2.2.3 Other architectures

This section describes other DL architectures useful in the field of computational biology and chemistry. Although they are not directly applied in this thesis, these architectures have been used to describe problems related to those of the scope of this thesis and thus, they have been included for completeness.

Attention mechanism

Attention mechanisms allow the network to focus only on a certain subset of the data provided for a given task (K. Xu et al., 2016). The idea behind these mechanisms is based in the visual attention of humans, who focus on specific parts of their visual inputs to compute adequate responses.

Being able to distinguish between the necessary information at a specific step of a task further reduces the amount of information that has to be processed. The analysis of where the model focus its attention can also help to understand the underlying structures and patterns of the information. The attentional mechanism has had considerable relevance in the field of neural machine translation (Bahdanau et al., 2014; Luong et al., 2015): when translating a word from one language to another, this mechanism locates those positions in which the most relevant information of the source sentence are concentrated (see Figure 14).

Attention mechanisms are usually used along with RNNs and CNNs (Bahdanau et al., 2014; K. Xu et al., 2016). In some cases, a CNN generates a hid-

4 https://colah.github.io/posts/2015-08-Understanding-LSTMs/

den representation that is fed to a RNN providing a description. Here, an attention mechanism allows to follow the information extraction and track the explicit spots of the input where the algorithm is looking for the answer (LeCun et al., 2015).

However, recently there have been introduced sequence to sequence mapping architectures which only use the attention mechanism without recurrence (RNNs or LSTMs) (Vaswani et al., 2017). Specifically, they use *self-attention*, an attention mechanism relating different positions of a single sequence in order to compute a representation of the sequence (Vaswani et al., 2017). The resulting architecture, called **Transformer**, surpasses any other NLP in both performance metrics and computational time (Devlin et al., 2018). Due to their popularity, Transformers are starting to be applied in many fields, including computational biology (J. Lee et al., 2019).

Unsupervised learning

As mentioned at the beginning of this section, one of the reasons of the recent resurgence of DL is the massive increase of available data. However, most of this information is unlabeled, i.e. there is only raw data without extra information on their identification, properties or classification. **Unsupervised learning** consists in the extraction of patterns from raw data and the discovery of its underlying structure at multiple levels. In contrast to supervised learning, unsupervised learning techniques capture all the possible dependencies between all the observed variables, with no distinction between inputs and outputs. This is more similar to how humans learn: extracting most of the information from simple observation. Down below, some of the DL techniques used for inferring features from unlabelled data are shown.

AUTOENCODERS

Autoencoders are a family of neural networks whose aim is to reproduce the input (Baldi, 2012). They work by compressing the input into a latent space representation, and then reconstructing the output from this representation (Figure 15). The model tries to minimize the reconstruction error between the input value and the reconstructed value. In order to learn useful features, constraints must be added to the network, so no neuron can learn the identity function but they will learn to project inputs in a lower dimensional space.

A common application of autoencoders is **dimensionality reduction**: after the training phase, building a new dataset of samples with lower dimensions. This extracted compressed representation can be used for statistical analysis of the data distribution, for a better classification or for understanding the underlying structure of data and recognizing possible patterns.

There are different modifications of autoencoders depending on the problem to which they are addressed, for example the **denoising autoencoder** (Vincent et al., 2008), which is trained to reconstruct the input from a corrupted version of it. This allows the autoencoder to discover more robust features than the ones that could be learned from the original uncorrupted

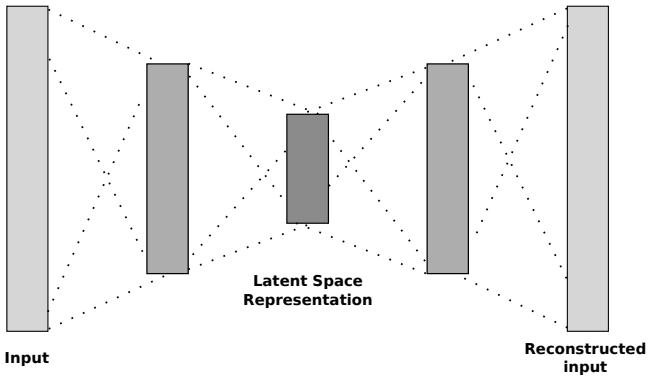

Figure 15: Simple Autoencoder architecture. It is composed by one input layer which *encodes* information and one output layer which *decodes* it.

data. The **variational autoencoders** (Diederik P. Kingma and Welling, 2014) assume that source data has an underlying probability distribution and then attempts to find the parameters of that distribution. Their main application is the generation of new data related to the original.

GENERATIVE ADVERSARIAL NETWORKS

Generative Adversarial Networks (GANs) (Goodfellow et al., 2014) consist of a given training set with an underlying distribution and two competing neural networks models: the *generative* model takes noise as input and generates samples, and the *discriminative* model receives samples from both the generative and the training data, having to be able to distinguish between the two sources, i.e. decide if they come from the same distribution. These two networks continuously compete: while the generative model is learning to produce increasingly realistic samples, the discriminative model is increasingly improving at distinguishing generated data from real data. Both networks are trained simultaneously until the generated samples are indistinguishable from real data (Lei et al., 2019).

Recent developments have extended the GANs to cover specific applications. *InfoGAN* (X. Chen et al., 2016) not only approximates the data distribution, but also generates an interpretable vector representation of the data. In *Conditional Generative Adversarial Networks*, the generative model takes into account external information such as the class label to generate a particular type of output (Isola et al., 2016; Ledig et al., 2016; Reed et al., 2016). *Deep Convolutional Generative Adversarial Networks* (Radford et al., 2016) combine the power of both types of architectures.

2.2.4 Quality assessment in deep learning models

DL models are usually referred to as "black boxes" in which it is difficult to interpret the reasons behind their prediction of a certain response. This lack of an straightforward interpretation is not only a problem for the adoption of these models by the professionals, as will be further addressed in section 2.5, but it also hinders the diagnosis of under performing models and leads

to a model development based on trial and error (Zeiler and Fergus, 2014). Here, the efforts to ensure the quality of DL models are grouped into two categories: those that make sure of the input data quality, and those that inspect the internal states of the DL model.

Input data quality

The database on which a ML model is trained can pose issues regarding its incompleteness, noise, imbalance and potential biases, which are further obfuscated by the black box nature of DL. This is something to take care of, since low quality data will lead to low quality knowledge (García et al., 2016) and thus, data preprocessing is an essential stage for building a DL model. Missing values imputation (Köse et al., 2020), noise treatment (Soltanayev and Chun, 2018), dimensionality reduction (Gang et al., 2018) or normalization (Ioffe and Szegedy, 2015) are examples of classical ML preprocessing techniques which have been also applied to DL models.

In particular high class imbalance, representative of many real-world settings, can lead to a bias of the learning model to the majority class. Effective classification with imbalanced data is an important area of research with little empirical work in the area of DL (J. M. Johnson and Taghi M Khoshgoftaar, 2019). Many of the available studies focus on computer vision tasks with CNNs (J. M. Johnson and Taghi M Khoshgoftaar, 2019). Some of the tools that have been used to address this problem are applied to the data level, for example random-oversampling or undersampling techniques (Buda et al., 2018; H. Lee et al., 2016; Masko and Hensman, 2015) or a dynamic sampling method that adjusts the sampling rate to the performance per class (Pouyanfar et al., 2018). Other methods address data imbalance by directly modifying the algorithm, for example through new loss functions (Ross and Dollár, 2017; Shoujin Wang et al., 2016) or cost-sensitive DNNs (Khan et al., 2017; Ren et al., 2018; H. Wang et al., 2018; C. Zhang et al., 2016). Other methods are hybrid, affecting both at the data-level and at the algorithm level (Dong et al., 2018; C. Huang et al., 2016).

Model inspection

On the other hand, a collection of strategies has been developed to better understand the behaviour of these complex models, being most of them domain-specific (Rudin, 2019). A comprehensive review of interpretation techniques for black box models can be found in Guidotti et al., 2018. One approach is to build a second, more explainable model, for diagnosing a DL model. However, due to the lack of transparency of the DL model and the inherent inaccuracy of the secondary interpretable model, some authors even discourage the use of black boxes for high stakes decisions and ask for the use of interpretable models instead (Rudin, 2019). While this might be necessary for highly sensitive settings like criminal justice or healthcare applications, not all the responses that need to be modelled can be represented by linear or additive algorithms. Thus, the use of interpretable models is not always possible in complex applications.

Alternatively, there are techniques that show what the model "sees" and the areas where the model is more sensitive, through the calculation and

Resource name	Main subject	Reference
ChEMBL	Compounds, bioactivity data	(Mendez et al., 2019)
BindingDB	Compounds, bioactivity data	(Gilson et al., 2016)
DUD-E	Decoys, bioactivity data	(Mysinger et al., 2012)
MUV	Unbiased bioactivity data	(Rohrer and Baumann, 2009)
DrugBank	Drugs, bioactivity data	(Wishart et al., 2018)
ZINC	Purchasable compounds, bioactivity data	(Sterling and Irwin, 2015)
UniProtKB	Proteins	(Consortium, 2018)
BRENDA	Enzymes	(Jeske et al., 2018)

Table 3: Selection of molecular databases for accomplishing the goal of chemogenomics.

analysis of extra outputs, such as activation values and gradients (Mehta et al., 2020). Besides, these techniques can be also used to evaluate the effect of changing the models training conditions (number of iterations, different initializations, model width, addition of normalization techniques, etc.) (Mehta et al., 2020). Nevertheless, these extra outputs scale as the product of sample size and number of parameters of the model and consequently, often imply a demanding computational framework (Mehta et al., 2020). Sometimes it is enough to analyse the *lower dimensional representation* learned by the last hidden layer for key samples of the input data (Mehta et al., 2020). For 2D CNNs there are specific tools to check these lower dimensional internal representations, like *network dissection* (Bau et al., 2017), which scores the semantics of the hidden units at each hidden convolutional layer and labels them with human interpretable concepts; or a visualization with a *deconvnet* (Zeiler and Fergus, 2014), which reveals the input stimuli that excite individual feature maps at any layer in the model. Other similar interpretability tools are Deep Visualization (Yosinski et al., 2015) or image-specific saliency maps (Simonyan et al., 2014).

2.3 MOLECULAR DATABASES

To get enough data to train statistical models and thus, to accomplish the goal of chemogenomics more effectively and systematically, it is necessary to have access to public compound libraries. These databases are composed of a structurally diverse collection of chemical molecules and information on their characteristics and their interaction to different biomolecules (specifically proteins). In most of databases, this protein-ligand affinity data has been extracted from scientific publications and subsequently curated. These binding activities need to be linked to biological entities in order to get their amino acid sequences and their correspoding annotations. Biological data repositories are freely accessible databases containing a large amount of information derived, among other things, from genome sequencing projects. A selection of public repositories within the scope of this thesis are listed in Table 3.

ChEMBL (Mendez et al., 2019) is an open-access, large-scale bioactivity database containing information manually extracted from the scientific literature and integrated with data on approved drugs and clinical development canditates. Biological activity data is also shared with other key public data-

bases as PubChem BioAssay (Y. Wang et al., 2017) or BindingDB (Gilson et al., 2016). ChEMBL covers a wide range of information on small molecules and has many pratical applications, like the identification of chemical tools for a target of interest, assessment of compound selectivity, training ML models, generation of drug repurposing hypotheses, integration with other drug discovery resources, etc (Mendez et al., 2019). ChEMBL is one of the most important bioactivity resources and there are available 15,996,368 activity records from 13,377 targets and 1,950,765 distinct compounds [5].

Another relevant resource of biological activity data is **BindingDB** (Gilson et al., 2016). BindingDB is a web-accessible database of measured binding affinities which collects information on interactions between proteins considered to be drug-targets and small molecules. Data collection also derives from scientific literature and, increasingly, from US patents. It has cross-links with other compounds catalogues. BindingDB currently contains about 1,854,767 binding data for 7,493 proteins and over 820,433 drug-like molecules [6].

One of the main problems of biological activity databases comes from the *publication bias* (Tripepi et al., 2008): journals are usually biased when deciding on accepting studies that report the lack of activty between a protein and a molecule and, alternatively, authors of negatives studies might not submit them because of the low probability for these to be accepted. Thus, there is a bias towards positive activity in public compound libraries which can affect to models trained with this data. The **Directory of Useful Decoys - Enhanced (DUD-E)** (Mysinger et al., 2012) gives means to overcome this problem. It is a database designed to help benchmark virtual screening programs. To build this directory, 22,886 active molecules for 102 proteins were derived from ChEMBL. For each of the actives, 50 decoys having similar physico-chemical properties but dissimilar 2-D topology were added to the database. These decoys are assumed to be inactive, thus effectively increasing the number of inactive interactions to train with.

The composition of the training datasets in virtual screening can also have a large influence on validation results. Rohrer and Baumann, 2009 warned of a *benchmark dataset bias* in virtual screening coming from two sources. On the one hand, the *artificial enrichment* caused by decoys and actives differing on "simple" physico-chemical properties, which was addressed by the DUD-E directory. On the other hand, the *analogue bias*, caused by datasets being prone to over-representation of certain scaffolds or chemical entities, thus inflating model performance. The **Maximum Unbiased Validation** (MUV) database (Rohrer and Baumann, 2009) aimed to minimize the influence of this *benchmark dataset bias*. It contains only 17 target proteins, each one represented by 30 active molecules selected from confirmatory screening data and 15,000 inactive molecules obtained from high-throughput screening data. The data is collected from PubChem bioactivity data (Kim et al., 2019). This benchmark is designed to be challenging for standard virtual screening, since the active molecules have been selected to avoid biases of enrichment assessments and the inactives have been biologically tested on the target protein (unlike DUD-E decoys).

5 https://www.ebi.ac.uk/chembl/. Accessed on 28/04/20.
6 https://www.bindingdb.org/. Accessed on 02/05/20.

The databases above collect information and bioactivity on small molecules. However, there are resources specifically focused on drugs. The most important drug information repository is **DrugBank** (Wishart et al., 2018). DrugBank is a large, comprehensive, drug data resource covering drug function, formulation, mechanism and metabolism. The database contains 13,570 drug entries, including among others, 2,629 approved small molecule drugs and 6,373 experimental (discovery-phase) drugs. Additionally, 5,252 non-redundant protein sequences are linked to these drug entries[7]. DrugBank is not only useful for model training but also for results contextualization.

Compound libraries are primarily focused on *in silico* activity prediction. But there is usually a gap between *in silico* obtained results and their experimental validation. **ZINC (Zinc Is Not Comercial)** (Sterling and Irwin, 2015) is yet another free database of compounds for virtual screening, whose distinctive feature is that it contains over 120 millions purchasable drug-like compounds. ZINC narrows the gap between biology and cheminformatics in both ways: it makes easier to purchase a compound predicted to be active in order to prove it experimentally and, additionally, it is also a toolset meant for investigators that are not computer specialists. Compounds are available in target-focused subsets. There are 3,210 gene symbols and 4,718 Uniprot codes (Sterling and Irwin, 2015).

Combining different sources of chemical structures can cause problems due to differing representations of molecules and even the use of a single source of data can lead to incorrect representation of compounds. For this, molecules must be standandardized and curated: 1. removal of inorganic molecules and mixtures; 2. structural conversion and cleaning; 3. normalization of specific chemotypes; 4. analysis and removal of duplicates and 5. manual inspection (if possible) (Fourches et al., 2010).

Up until this moment, only compound bioactivity libraries have been mentioned. These ligand databases have annotations on associated proteins and genes for each molecule, but not specific information or characteristics of these sequences is provided. This information is essential, not only for understanding data and interpreting results, but also for improving the protein analysis blocks on chemogenomics models. **The UniProt Knowledgebase (UniProtKB)** (Consortium, 2018) is the most comprehensive resource for protein sequences and annotations. It has two sections: *UniProtKB/Swiss-Prot*, with 562,253 manually-annotated records of information from literature evaluated by curators, and *UniProtKB/TrEMBL*, with 180,690,447 computationally analyzed records not yet reviewed [8]. UniProtKB stores the amino acid sequence, protein name or description, taxonomic data, citation, widely accepted biological ontologies, classifications and cross-references, and indications of the quality of annotation of each protein.

Among biological macromolecules that can potentially be drug targets, enzymes have a prominent position because their activity is essential in many disease processes, and because the structural determinants of enzyme catalysis lend themselves well to inhibition by small compounds (Krause et al., 2013). That is why the **BRENDA Enzyme Database** (Jeske et al., 2018) can also be of interest in the context of this thesis. BRENDA is the main enzyme

7 https://www.drugbank.ca/. Accessed on 03/05/20.
8 https://www.uniprot.org/. Accessed on 04/05/20.

and enzyme-ligand information system, with a collection of 4.3 million data for 84,000 enzymes from the literature and prediction algorithms. It contains disease-related data, protein sequences, 3D structures, predicted enzyme locations and genome annotations.

2.4 DEEP LEARNING IN COMPUTATIONAL BIOCHEMISTRY

In previous sections, the transformative impact of DL in most domains has been shown along with an overview of its capabilities, shortcomings and applications. This section will focus on the advancements of DL techniques in the bio- and cheminformatics fields.

2.4.1 Application to biological sequences

Biological sequences play a major role in molecular and computational biochemistry. They are studied as information-bearing entities that make up DNA, RNA or proteins. The value of DNNs in this context is twofold (Angermueller et al., 2016): on the one hand, classical ML methods cannot operate on the sequence directly; they require features extracted from the sequence based on prior knowledge, while DNNs could **directly learn** them from data. On the other hand, because of their **representational potential**, they can capture nonlinear dependencies in the sequence and interaction effects, spanning wider sequence context at multiple genomic scales. DL has the potential to provide additional understanding about the structure of the biological data.

DL has been used in many applications in the field of biological sequences, such as regulatory biology, variant calling and mutation detection, protein annotation, transcriptomics, etc (Crawford and Greene, 2020). In the following section we will see specific problems tackled with DL in this context, along with the most popular architectures for solving them.

Deep learning architectures

In 1988, the first application of neural networks in biological sequences was published, consisting in protein secondary structure prediction by a feed-forward neural network. Since feed-forward neural networks cannot handle sequential input, sequences had to be divided in windows of 13 amino acids (Qian and Sejnowski, 1988). After the breakthrough of DL, this issue was further addressed (Spencer et al., 2015) but still not treating the sequence as a whole, but windowing it as input of basic feed-forward networks. More recently, feed-forward neural networks have been also applied to infer expression from target genes (Min et al., 2017).

Biological sequential data often contain correlated measurements of related biological activities (Eraslan et al., 2019). So instead of building a single-task model for predicting each of those activities, a *multi-task model* can be used. Based on this and with a more ambitious approach than Qian and Sejnowski, 1988, Qi et al., 2012 built a multi-task DNN model based on feed-forward neural networks that given a protein sequence, predicted

different local properties such as secondary structure, solvent accessibility, transmembrane topology or DNA-binding residues. In multi-task models, the majority of layers are shared and branch out to task-specific layers at the end (Eraslan et al., 2019). The advantages of this strategy are that less specific data per task is needed and that sharing the model implies less computation time and faster predictions. However, it can also be challenging if different tasks have different data imbalance. In the most recent literature we can still see this strategy: in Rifaioglu et al., 2019, a stack of multi-task feed-forward neural networks connected to each other is used to predict Gene Ontology terms of a given protein.

However, these previous works did not take into account the specific characteristics of biological sequences. The appropriate DL architecture to manage this kind of data is **RNN**, because biological sequences have variable lengths and their long-range sequential information has great importance. RNNs apply the same operation to each element, thus they are invariant to the position index in the processed sequence (Eraslan et al., 2019).

The group of Baldi et al was one of the most prolific regarding the first applications of RNNs to protein sequences. In Baldi et al., 1999, they proposed a model for predicting the secondary structure of a protein as seen before, but using the whole protein sequence rather than a short substring by means of RNNs. In addition, they realized that RNNs are *causal* in the sense that the output at some time point does not depend on future inputs, but biological sequences are not temporal since the conformation and function of a region in a sequence may strongly depend on events located both upstream and downstream. To tackle this issue they developed a bi-directional structure that provided a non-causal generalization of RNNs.

Similar approaches with different improvements were applied to match protein beta-sheet partners (Baldi et al., 2000), to predict protein structure using directed acyclic graphs (Baldi and Pollastri, 2003), to predict residue-residue contact (Di Lena et al., 2012), to classify proteins (Min et al., 2017) and again, to predict protein secondary structure but identifying the effect of the contribution of local versus global information (Agathocleous et al., 2010). In the previous section it was shown that classical RNN cannot hold very long-range dependencies. However, in these studies the important information in the sequence was close to the prediction position.

To avoid the problems of vanishing and exploding gradients of classical RNNs, Hochreiter et al., 2007 applied **LSTMs** to classify amino acid sequences into superfamilies. This model reached state-of-the-art classification performance, but being considerably faster than other methods based in sequence alignment. Moreover, LSTM was also able to extract characteristic patterns of the proteins. But the LSTM used was one-directional, so it did not consider the non-causal nature of amino acid sequences. A novel application of LSTMs in the protein context was carried by Müller et al., 2018. They used LSTM for combinatorial *de novo* peptide design. In this case, LSTM was also one-directional, but because in sequence generation the amino acid sequence is unknown *a priori*.

Sønderby and colleagues did actually consider non-causality, and used a **bidirectional LSTM** to address the problem of prediction of protein secondary structure (Kaae Sönderby and Winther, 2015). They also presented a

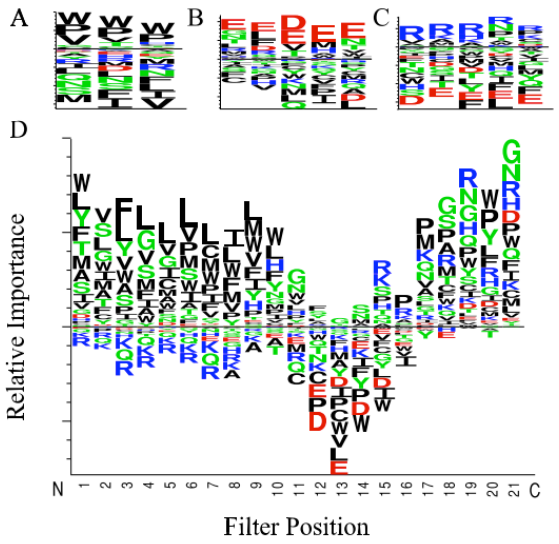

Figure 16: Examples of convolutional learned filters from Sönderby et al., 2015. Shorter filters (A, B, C) focus on amino acids with specific characteristics, such as positively or negatively charged. The long filter (D) seems to focus on larger motifs from the protein. Figure permissions granted by the open access Creative Common CC BY license.

more complex architecture composed of a convolutional layer, whose learned filters can be used to discover new sequence motifs (see Figure 16); two combined unidirectional LSTMs, one of them with the input reversed, instead of a regular bidirectional LSTM to avoid that the backward LSTM sees only a single position when the prediction is made; and also an attention mechanism to see the importance of each position in the sequence with respect to the prediction task. This architecture was also used by the same group to predict subcellular locations of proteins (Almagro Armenteros et al., 2017) and binding to major histocompatibility class II molecules (Jurtz et al., 2017). A similar architecture was used by Quang and Xie, 2016, in this case to build an hybrid convolutional and bi-directional LSTM framework for predicting non-coding function *de novo* from sequence. Convolutional layers and a unidirectional LSTM were also combined for a hierarchical classification of enzymes by Yu Li et al., 2018.

In genomics, RNNs have been used to aggregate the outputs of the convolutional layers for predicting RNA-binding proteins junction sites (Pan et al., 2018), gene expression regulation (Min et al., 2017), single-cell DNA methylation states, transcription factor binding and DNA accesibility (Eraslan et al., 2019). It has also been widely applied for miRNAs sequences: prediction of binding mRNA-miRNA sequence pairs and prediction of the occurrence of precursor miRNAs from the mRNA sequence and the predicted secondary structure (Eraslan et al., 2019), among others.

As shown in Figure 16, the application of **1D CNNs** on biological sequential data is also very useful: the neurons in the convolutional layer scan for *motif sequences* and combinations. One the first examples of this approach can be found in DeepBind (Alipanahi et al., 2015), a software tool based on CNNs to predict DNA- and RNA-binding protein sequences specificities.

Since this first application, CNNs have been applied to predict various molecular phenotypes on the DNA sequences alone: classification of transcription factor binding sites, genetic variant calling or prediction of molecular phenotypes such as chromatin features, DNA contact maps, DNA methylation, gene expression, translation efficiency, or microRNA targets. (Eraslan et al., 2019). CNNs have been also applied for predicting diffraction quality crystals from non-crystallizable ones from the protein sequences (Elbasir et al., 2019) or for predicting the probability of a short sequence of DNA being evolutionary conserved (Yi Li et al., 2017). Torng and Altman, 2017 used 3D-CNNs to analyze amino acid micro-environment similarities and to predict effects of mutations in proteins.

Moreover, CNNs (or, specifically, *dilated convolutions*, which allow for bigger receptive fields) can also be used to model long-range dependencies in biological sequences, because interacting elements that are proximal in the 3D conformations of the sequence may be distantly located on the unfolded sequence (Eraslan et al., 2019). These dilated convolutions have been used for predicting trascription factor-binding profiles from DNA sequences (Crawford and Greene, 2020).

Convolutional layers of different filter sizes can be interpreted in biological sequences as multiple position weight matrices (PWMs) (Ben-Gal et al., 2005) scanning the sequences, although complete motifs can only be detected by assembling multiple filters in the downstream layers (Eraslan et al., 2019). Since DNN learn distributed representations, the patterns recognized by individual filters are often found to be partially redundant with each other (Shrikumar et al., 2018). TF-MoDISco is a method that identify consolidated motifs learned by the DL model using information from all the neurons in the network (Shrikumar et al., 2018), being one of their main target applications DNA sequences (Avsec et al., 2020). However, these learned motifs are based on the overall trained model, but one may be only interested in highlighting the influential parts of a given input for a model prediction (*feature importance scores*) (Eraslan et al., 2019). DeepLIFT (Shrikumar et al., 2017) is a method for decomposing the output prediction of a NN on a specific input by backpropagating the contributions of all neurons in the network to every feature of the input; it was tested in a synthetic regulatory DNA sequence classification.

Multi-task learning has been previously mentioned as a way of training a single model for performing a certain number of prediction tasks. In genomics, multi-task models have been used for predicting simultaneously multiple molecular phenotypes (Eraslan et al., 2019) or chromatin factors (Min et al., 2017). However, a similar approach when data is scarce is *transfer learning*, where a model is initialized with most of the parameters from another model which has been trained for a similar task (Eraslan et al., 2019). In genomics, transfer learning has been used for predictive models of chromatin accesibility (Kelley et al., 2016).

Most of the previous examples referred to applications of supervised learning for biological sequence processing. But unsupervised methods have also been increasingly applied to this kind of data. Autoencoders have been used to extract gene expression signatures, to impute missing data or to detect expression outliers in gene expression data (Eraslan et al., 2019). Stacked

autoencoders have been also used for predicting secondary structure and accessible surface area and torsion angles in proteins, and for anomaly classification of various cancers from microarray gene expression data (Min et al., 2017). GANs have been used for generating protein-coding DNA sequences and for designing DNA probes for protein binding microarrays (Eraslan et al., 2019). A variational autoencoder has been used for building a deep generative model able to capture higher-order correlations in biological sequence families. It has been applied for predicting the effects of mutations across diverse classes of proteins and RNAs. (Riesselman et al., 2018).

Automated tools are being developed for building DL models for biological proteins. For example, SECLAF is a web-server that allows for hierarchical biological sequence classification using deep neural networks; it has been used for GO classes and protein families classification (Szalkai and Grolmusz, 2018). However, despite the numerous applications of DL to biological sequences, there is a lack of systematic comparison of the different architectures for the common sequence-modelling tasks (Eraslan et al., 2019).

Input representations

DNA and RNA nucleotides or amino acids are categorical features. Specifically, DNA and RNA are polymers constructed from four different types of nucleotides: {A, T, C, G} and {A, U, C, G}, respectively, and proteins are polymers made up of 20 different types of amino acids {A, C, D, E, F, G, H, I, K, L, M, N, P, Q, R, S, T, V, W, Y}. In both cases, elements are represented by alphabet characters, although A, T, G and C have a different meaning as a nucleotide than as an amino acid. Thus, they have to be numerically encoded in order to be be interpretable by machines (Angermueller et al., 2016) (see section 1.5). Finding the proper data representation for each task is very important because it can have an impact on the performance of the model (Domingos, 2012). The core idea is to infer information about structure/functions only based on sequence data (Asgari and Mofrad, 2019). In the previous section it was shown that in the first implementations of DL in biological sequences, input sequences were windowed (Qian and Sejnowski, 1988) or amino acids were directly embedded without taking into account contextual information (Qi et al., 2012). Since then, the input representation field has advanced substantially. In this section, different input representations for biological sequences are covered: from those more traditional to the ones that avoid losing sequential information (see Figure 17).

Traditional representations of biological sequences mainly include alignment-dependent descriptors and vectors of physico-chemical features, as seen in section 1.5. In Jing et al., 2019, a comprehensive review and assessment on different amino acid encoding methods was performed, showing that position specific scoring matrix (PSSM), an evolution-based position dependent methodology, achieves the best performance. POSSUM (J. Wang et al., 2017) is a popular tool for generating PSSM-based descriptors, although they may lack the evolutionary information for more specific tasks as modelling DNA or RNA-binding sites (L. Wang et al., 2010). Moreover, it is a computationally consuming method (Ahmad and Sarai, 2005) and its appli-

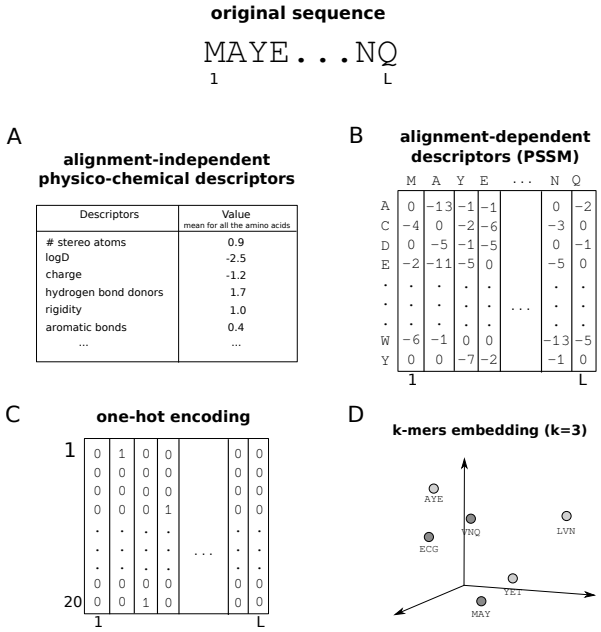

Figure 17: Main input representations for biological sequences, with an amino acid sequence as an example. **A.** alignment-independent physico-chemical descriptors, **B.** alignment-dependent descriptors (in the example, PSSM stands for position specific scoring matrix), **C.** one-hot encoding, **D.** embeddings based on sequence k-mers (in the example, k=3)

cability is limited to those proteins with known homologous sequences (Jing et al., 2019).

Regarding feature vectors, different properties such as a nucleotide/amino acid composition, evolutionary signatures or bio- and physicochemical properties (Z-scales) have been usually aggregated and used as input. Several tools have been developed to compute these properties and signatures from a DNA/RNA sequence, like repDNA (Liu et al., 2014) or from a protein, like PROFEAT (P. Zhang et al., 2017) or BindN+ (L. Wang et al., 2010). There are also tools that compute properties for both DNA/RNA and amino acid sequences, like BioSeq-Analysis (Liu et al., 2019) or iLearn (Z. Chen et al., 2019). These feature vectors have been used to represent biological sequences in a wide range of studies (Lenselink et al., 2017; Meysman et al., 2012; Strömbergsson et al., 2010). The main issue with these descriptors is that the original sequence is lost. This means that the underlying information related to the position of the elements may be neglected.

Pseudo amino acid composition (PseAAC) (H.-B. Shen and K.-C. Chou, 2008) and pseudo nucleotide composition (PseKNC) (Zhou et al., 2011) formulations tried to overcome this issue. PseAAC and PseKNC are a series of correlation factors introduced along the sequence to approximately reflect the sequence order effect. These descriptors became very popular for classical ML methods, which cannot handle sequential inputs (W. Chen et al., 2015). However, since DL models are able to operate directly on the sequence, summarizing or approximating sequence information is not required anymore.

There is an obvious analogy between natural languages (such as English, Spanish or Catalan) and the *language of life* of biological sequences (that of DNA, RNA or proteins). A natural language is a infinite set of sentences composed from a finite set of words, while the macromolecules of life are unlimited polymers constructed from a certain set of smaller molecules (Asgari and Mofrad, 2019). Similar to the syntactic and semantic rules of natural languages, biophysical and biochemical grammars guide the formation of biological sequences (Asgari and Mofrad, 2019). Thus, classical NLP methods have been adopted in this context to get a deeper understanding of how information is encoded within biological sequences (Asgari and Mofrad, 2019). Specifically, one-hot encoding (Alipanahi et al., 2015; Jurtz et al., 2017; Kelley et al., 2016; Yu Li et al., 2018; Lovino et al., 2019; Müller et al., 2018; Öztürk et al., 2018a; Pan et al., 2018; Quang and Xie, 2016; Rifaioglu et al., 2019) and machine-learning derived embeddings (Asgari et al., 2015; Asgari and Mofrad, 2019; Kimothi et al., 2016; Mazzaferro, 2017; Ng, 2017; Z. Shen et al., 2018; Trabelsi et al., 2019; K. K. Yang et al., 2018) have become very popular.

In one-hot encoding, each position of the sequence is represented by a binary vector with all but one entries set to zero, indicating this one its corresponding element in the alphabet. Formally, a biological sequence of length L can be encoded as an $L \times n$, matrix, where n is the number of element categories (4 in the case of RNA/DNA, 20 in the case of protein sequences). Each row in the matrix represents an element in the sequence and consists of $(n-1)$ zeros and a single one, indicating which element of the alphabet it is according to its position. One-hot encoding does not lose sequence information because given a position, its element can be recovered unambiguously, without needing to extract features based on previous knowledge. The main problems of one-hot encoding are that it is sparse, memory-inefficient, high-dimensional and lacks of a notion of similarity between sequence or structural elements: they are either identical, or not (K. K. Yang et al., 2018).

Moreover, biological sequences differ in length, but all input vectors have to be of the same size to be processed by a DL model. Researchers usually fix this by establishing a common length for all the sequences (Jurtz et al., 2017; Lovino et al., 2019; Müller et al., 2018; Öztürk et al., 2018a; Rifaioglu et al., 2019). Sequences longer than this threshold are *truncated*, i.e. all the elements from the limit length on are discarded. Shorter sequences are *padded*, this is, they are filled with some pre-selected character (usually 0) up until the established common length; these padding elements can be added at any point of the sequence.

On the other hand, embedding biological sequences in a lower dimensional vector space of fixed-length makes the data more suitable for ML tools, as long as the quality of the embedding is high and captures the most meaningful information of the original sequences (Kimothi et al., 2016). However, getting a quality embedding requires training it with a large corpus of sequences to ensure sufficient contexts are observed (Asgari et al., 2015). This is not always possible, because of both computational and availability reasons. Increasingly, biological pre-trained embeddings are being published (Asgari et al., 2015; K. K. Yang et al., 2018). However, the training of these embeddings still have some limitations: in K. K. Yang et al., 2018, their em-

bedding is trained with UniprotKB sequences shorter than 1000 amino acids. In Asgari et al., 2015, their embedding is trained with all the annotated proteins in UniProtKB/Swiss-Prot, neglecting all the unlabeled data available in UniProtKB/TrEMBL. Up to our knowledge, to this date there are still no pre-trained embeddings for DNA/RNA sequences. Embeddings have great potential for encoding the input in biological sequences DL models, but there is still room for improvement. Additionally, embedding vectors can only partially recover input data (Song and Raghunathan, 2020).

Recent efforts are also exploring the representation of biological sequences using pre-trained self-attention structures (Transformers). The application of Transformers in computational biology is still very recent (J. Lee et al., 2019). Although there have been some approaches for creating a learned representation space of proteins based on Transformers (Rao et al., 2019; Rives et al., 2019), no large scale protein representation is yet publicly available. This limits their adoption by the researcher community.

2.4.2 Application to cheminformatics

As seen before, QSAR and QMSPR models have been traditionally implemented with ML algorithms, especially Random Forests (RF) (Breiman, 2001), SVMs (Cortes and Vapnik, 1995) and Extreme Gradient Boosting (T. Chen et al., 2015), which are among the most predictive methods (Bruce et al., 2007; Sheridan et al., 2016; Svetnik et al., 2005). A general task for ML is to infer the relationship between the molecular descriptors used and the measured activity of the compounds. However, these *shallow* ML models have some issues: first, a manual design of the information that should be provided as the input of the ML model must be carried out by a domain expert in a difficult, time-consuming process called *feature engineering*; second, traditional ML does not learn a representation of the problem, but it simply learns how to balance in a precise way a set of input features to produce an output; and third, its expressive power grows with the number of parameters to be fitted, and these parameters may grow exponentially if the nonlinear transformations are wrongly chosen (Goh et al., 2017b).

As it was seen in previous sections, DL is able to map features through a complex, optimally combined series of nonlinear functions, learning multiple levels of abstract representations. These **hierarchical representations** are kept in hidden units, constituting an internal state for the DL model that may be transferred to new problems. This is important in multi-task learning, especially for those targets with very few measurements available. In addition, in DL expressive power grows exponentially with depth (Goh et al., 2017b). These characteristics allow to use raw data directly and therefore, make DL appealing to a field with such complex data as it is cheminformatics.

Early work in QSAR applications used linear regression models, but these were quickly supplanted by Bayesian neural networks, followed by RFs and SVMs (Goh et al., 2017b). In 2012, Merck launched a QSAR ML challenge on drug discovery and activity prediction [9]. The competition was won by multi-task neural networks which outperformed Merck's in-house systems

9 Merck Activity Competition, 2012: https://www.kaggle.com/c/MerckActivity

with a relative accuracy improvement of approximately 14% (Dahl et al., 2014). 3764 molecular descriptors per molecule were generated as input features for the neural network, developing a shared, learned feature extraction pipeline for the prediction of the different assays of the competition. The winning team had not any chemist or biologist among them. In a subsequent study published by Merck, DNNs were shown to outperform RF as QSAR method in most proven cases, with architectures of two hidden layers, 500-1000 neurons per layer and 75 training epochs (Ma et al., 2015). Nowadays, DL is an standard method in cheminformatics applications. In the following section we will show some DL strategies for solving different cheminformatics problems.

Deep learning architectures

Neural networks began to gain importance in the field of CADD during the nineties, thanks to their capability of recognising patterns, which was used for predicting the mechanisms of action in a cancer drug screening program and for fully automating a molecular design method (X. Yang et al., 2019). However, these were isolated developments and, as seen in the introduction of this section, the breakthrough of contemporary DL took place in the 2010s decade.

Multi-task learning is an strategy that has been vastly used since the first applications of DL to cheminformatics. As stated in section 2.4.1, the use of multi-task DNNs provides two main advantages: it allows for multi-label information and for sharing learned representations among prediction tasks (Goh et al., 2017b). In 2014, an architecture of multi-task fully-connected layers (similar to the one that won the Merck challenge (Dahl et al., 2014)) was applied for QSAR prediction on the entire curated ChEMBL database, outperforming all the ML models with which they were compared (Unterthiner et al., 2014). A year later, Ramsundar et al., 2015 trained the same architecture with nearly 40 million experimental mesurements of binding affinities to investigate the advantages of multi-task learning respect to single-task learning. A similar comparison between multi-task and single-task learning but only for kinases was recently carried, obtaining comparable results (Rodriguez-Perez and Bajorath, 2019). Multi-task feed-forward neural networks have been also found to perform better than single-task NNs and classical ML methods in a proteochemometrics context (Lenselink et al., 2017).

In all the previous examples, feed-forward neural networks were used to predict binding affinities in a QSAR setting. But in the cheminformatics field, **convolutional architectures** can also be applied to look for hidden representations. For example, in DeepCCI (Kwon and Yoon, 2017), 1D CNNs are applied to predict chemical-chemical interactions from SMILES strings. In DeepDTA, drug-kinase binding affinity is predicted from protein and compounds sequence information modeled with 1D CNNs (Öztürk et al., 2018a). In DeepConv-DTI, convolutional filters were applied to the entire sequence of a protein to capture local residue patterns which were later concatenated with drug features to predict drug-target interactions (Id et al., 2019). Duvenaud et al., 2015 extended the convolutional concept to molecules represented as 2D molecular graphs for extracting molecular rep-

resentation. A different approach is taken in Gomes et al., 2017; Ragoza et al., 2017; Stepniewska-Dziubinska et al., 2018; Wallach et al., 2015, where **3D convolutional layers** are applied to grid representations of protein-ligand interactions. In Ragoza et al., 2017, the aim is defining a scoring function to evaluate binding poses, and in Gomes et al., 2017; Stepniewska-Dziubinska et al., 2018; Wallach et al., 2015 they attempt to predict bioactivity.

Due to the sequential nature of some of the most used descriptors, **recurrent networks** are also very useful in the cheminformatics context. For example, LSTMs have been used for processing SMILES representations of compounds (Bjerrum, 2017; Mayr et al., 2018) and they have been stacked for predicting drug-target interaction in a proteochemometrics context (Y. B. Wang et al., 2020). The architecture that combines convolutional and recurrent layers and attention mechanisms has been also used in cheminformatics. In section 2.4.1 we mentioned it for predicting peptides binding to major histocompatibility class II molecules (Jurtz et al., 2017). In DeepAffinity (Karimi et al., 2019), they combined RNN-based autoencoders for input representation with the 1D CNNs-RNNs and attention mechanisms for compound-protein affinity prediction.

The Pande Group developed an open-source Python library called *DeepChem* with TensorFlow implementations with the aim of democratizing the use of DL in drug discovery (Ramsundar, 2016b). This tool was used to carry out different analysis like comparing multi-task deep networks to random forests (Ramsundar et al., 2017) or one-shot learning with recurrent LSTM (Altae-Tran et al., 2017), both for bioactivity prediction. *One-shot learning* is a technique to improve predictive power for tasks with sparse data, allowing non-trivial predictors to be learned from only a few datapoints (Altae-Tran et al., 2017).

Despite most of the studies focusing on biological activity prediction, DNNs have been used to predict other properties of interest in CADD like the site of reactivity of small molecules (Hughes et al., 2015b), molecule solubility through RNNs (Lusci et al., 2013), formation of reactive epoxidation and its location (Goh et al., 2017b) or all ADMET (Absorption, Distribution, Metabolism, Excretion-Toxicity) properties (Goh et al., 2017b; Hughes et al., 2015a; Kearnes et al., 2016; Korotcov et al., 2017; Mayr et al., 2016; Y. Xu et al., 2015; X. Yang et al., 2019). In Goh et al., 2017c, CNNs are used for the prediction of chemical properties (solvation, toxicity, activity). Aliper et al., 2016 classified drugs into therapeutic categories with feed-forward NNs.

Regarding the general architecture of the models, Koutsoukas et al., 2017 showed that DL models are robust to an arbitrary amount of noise if the number of "proper" samples is sufficiently large. Moreover, they observed that certain hyperparameters greatly affect the performance of DNN models, including the activation function, dropout regularization, number of hidden layers, and number of neurons per hidden layer. Other innovative approaches that have been taken in DL-based cheminformatics are deep belief networks to predict drug-target interactions (Ghasemi et al., 2017; Wen et al., 2017); stacked autoencoders for predicting drug-target interactions (Hamanaka et al., 2017; Y. Wang et al., 2017); or graph convolutional networks applied to 2D molecular graph analysis (Y.-c. Lo et al., 2018).

One of the most prolific applications of DL in CADD nowadays is the development of generators for reasonable *de novo* molecules. Different techniques such as GANs, reinforcement learning (a goal-oriented ML paradigm in which models are able to learn from their own errors) or generative adversarial autoencoders have been used to create new compounds whose features are within a pre-defined range (Born et al., 2020; Kadurin et al., 2017a; Neil et al., 2018; Putin et al., 2018; Sanchez-Lengeling et al., 2017). Sometimes these architectures were combined with recurrent networks in order to learn the long-range dependencies of the SMILES representation (Hs Segler et al., 2017; X. Yang et al., 2019). The ability of RNN-based generative approaches to suggest molecules with similar biochemical activities to those of a template but with novel scaffolds has been the subject of much interest (X. Yang et al., 2019).

Input representations

Although DL is able to extract abstract representations of input data, the model has to be fed with some kind of information in the first place. As seen in the Introduction and in section 2.4.1, thousands of descriptors can be computed in order to represent a molecule, but only some of them are chosen for reasons of simplicity, computational speed, storage and to avoid overfitting. In section 1.5, the main types of molecular descriptors were presented (see Figure 5).

Traditionally, the most frequent data representation in drug discovery is a fixed-length input vector of properties derived from the molecular representation of the compound (Altae-Tran et al., 2017; Dahl et al., 2014; Mayr et al., 2018; Ramsundar et al., 2017). The simple physicochemical properties of a small-molecule drug candidate have a well-known impact (X. Yang et al., 2019). However, in general, substructure descriptors (e.g., atom pairs, fingerprint descriptors (Todeschini and Consonni, 2008) seem to be more generally useful than descriptors that apply to the whole molecule (e.g., number of donors, molecular weight or logP) (Ma et al., 2015). In molecular 2D fingerprints, structural features are represented by either bits in a bit string or counts in a count vector, which makes them computationally efficient. There are four classes of algorithms to construct 2D fingerprints: 1. dictionary-based, 2. topological or path-based, 3. circular and 4. pharmacophores. Fingerprints have shown strong correlation and similar performance even when belonging to different classes (Riniker and G. A. Landrum, 2013), although topological torsions fingerprint was the only one ranked among the top by all evaluation methods in a study comparing performance of different fingerprints (Riniker and G. A. Landrum, 2013). In Unterthiner et al., 2014, Extended Connectivity Fingerprints were used to feed the model, representing chemical substructures of the compounds in the first layer, with the aim of forming pharmacophores (sets of steric and electronic properties of compounds that together enable an interaction with a target) in higher layers. The same fingerprints were used in Ramsundar et al., 2015. Other examples of fingerprints used as model input can be seen in Kadurin et al., 2017a; Korotcov et al., 2017; Koutsoukas et al., 2017; Mayr et al., 2016; Rodriguez-Perez and Bajorath, 2019. It used to be thought that longer fingerprints performed

better than shorter ones (Sastry et al., 2010), but later studies showed small effect of the length of the bit string (Riniker and G. A. Landrum, 2013).

There are many software tools for generating descriptors from molecular structures. The most popular is *RDKit*[10], an open source tool for cheminformatics with wrappers for Python, Java and C# able to build a wide range of descriptors and fingerprints[11]. Other well-known resources for building molecular descriptors for machine learning are DRAGON (Mauri et al., 2006), which is a comercial software for the calculation of 4885 molecular descriptors[12] and OpenBabel, an open chemical toolbox that allows, among many utilities, to compute descriptors[13]. It also has a Python implementation, Pybel (O'Boyle et al., 2008).

One of the major limitations derived from this type of representations is that such vectors tend to be quite large to encode all possible substructures without collisions, resulting in models with many learnable parameters attempting to learn from relatively sparse inputs (X. Yang et al., 2019). Additionally, they are sparse (only around 5% of the bits being nonzero) and strong correlations can exist between different fingerprints (Ma et al., 2015). Moreover, these representations do not allow for univoquely retrieving the original compound, which derives in the consequent loss of information.

Instead of fixed-length vectors of properties, several studies (Jastrzebski et al., 2018; Kwon and Yoon, 2017; Öztürk et al., 2018a) used as input a different type of sequence: the SMILES of the molecule, which is a string notation representing its chemical structure (Weininger, 1988). This type of representation has become especially popular in *de novo* molecular generation applications (Born et al., 2020; Hs Segler et al., 2017; Neil et al., 2018; Putin et al., 2018; Sanchez-Lengeling et al., 2017). SMILES are usually one-hot encoded and padded to be fed into the model (Hs Segler et al., 2017; Kwon and Yoon, 2017; Öztürk et al., 2018a; Sanchez-Lengeling et al., 2017). Bjerrum, 2017 showed that adding multiple SMILES representations (by scrambling randomly the atom ordering of the molecule) to augment the dataset improved QSAR models performance.

There are also more ambitious drug-target interaction prediction models that take advantage of the 3D structure of the binding site, taking as input vectorized versions of 3D grids placed within the target's binding site (Ragoza et al., 2017; Stepniewska-Dziubinska et al., 2018; Wallach et al., 2015). However, these are very demanding computationally and this kind of information is not available for all drug-proteins pairs. 2D drawings of molecules (Goh et al., 2017c) have been also used as model inputs.

NLP methods have also influenced studies on novel ways of encoding compounds. One example is SMILESVec (Öztürk et al., 2018b), a SMILES-based methods to represent ligands. In this study, the proteins are defined by the word-embeddings of the SMILES strings of their ligands. Another example is Goh et al., 2017a, a RNN-based SMILES embedding. Self atten-

10 RDKit: Open-source cheminformatics: http://www.rdkit.org

11 List of available descriptors and fingerprints in RDKit: https://www.rdkit.org/docs/GettingStartedInPython.html#list-of-available-descriptors

12 List of molecular descriptors computed by Dragon: http://www.talete.mi.it/products/dragon_molecular_descriptor_list.pdf

13 List of OpenBabel descriptors: https://open-babel.readthedocs.io/en/latest/Descriptors/descriptors.html

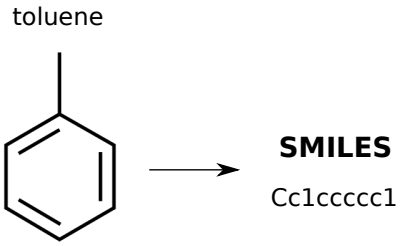

Figure 18: The molecule toluene and its corresponding SMILES. In contrast to upper case C, lower case C indicates aromatic atoms. Since it is a ring structure, number 1 indicates closure labels to show connectivity between non-adjacent atoms.

tion (Transformers)-based embeddings are also starting to be developed for molecules. Some examples of this trend can be seen in Honda et al., 2019; Qiu et al., 2017; Sheng Wang et al., 2019. However, as in the case of biological sequences, the lack of available public resources limits a higher adoption of this encoding by the community.

In addition to the molecular representations discussed above, proteochemometrics models also need to encode proteins. Different combinations of molecular descriptors and protein representations (as seen in section 2.4.1) can be found in the literature (Hamanaka et al., 2017; Id et al., 2019; Karimi et al., 2019; Wan and Zeng, 2016; Y. B. Wang et al., 2020; Y. Wang et al., 2017; Wen et al., 2017). Specifically, some of the studies compared the performance of different descriptor combinations (Lenselink et al., 2017; Öztürk et al., 2018a).

2.4.3 Validation schemas

Only a small portion of the chemical space has ever been synthesized and thus, research efforts in cheminformatics cluster near the current known working molecules (Wallach and Heifets, 2018). As stated in Ing, 1964, it is a traditional practice in drug discovery research to take a compound which is known to possess some desirable pharmacological properties as a model to study the effect of making small changes in its structure. The main advantage of these studies focused on homologous series is the ease and relative cheapness of preparating compounds with such small changes involved between them.

However, this could entail an obstacle for a machine learning model trained for classifying molecules as active or inactive against some target. If there are compounds with high structural similarity and they are *randomly spread* between the training and test sets, over-optimistic results will likely be obtained when evaluating the model. On this matter, some studies suggested that the performance of most of the reported ligand-based prediction methods are due to overfitting to training data rather than to good prospective accuracy (Kearnes et al., 2017; Wallach and Heifets, 2018); in Wallach and

Heifets, 2018, they defined a bias measurement for ligand-based benchmarks and an algorithm to reduce redundancy bias in order to solve it.

Transfer learning was explained in section 2.4.1: a model uses the parameters from another model which has been trained for a similar task. It is used when data is scarce or when working with different datasets. Transfer learning can be also a way of checking if the predictive power of a model is kept when applied to data from a different source. For example, limited transferability was observed in Ramsundar et al., 2015 when testing models in a database different to that in which it was trained. The same issue is found in Altae-Tran et al., 2017, where transfer learning experiments collapsed, leading to affirm that one-shot learning methods may struggle to generalize to novel molecular scaffolds. This hints that there could be bias across different data sources.

Due to these facts, the validation strategy used is crucial for proving a cheminformatics model robust, consistent and reliable. Data redundancy must be acknowledged and it should be tested if predictive power remains stable when applied to data that comes from different sources. In this section, different approaches that have been taken for model validation will be addressed.

One solution for preventing differences when evaluating drug-target interaction prediction models is having a **common setting for comparing results**. This approach was taken by Riniker and G. A. Landrum, 2013, who created a benchmarking platform with selected targets and their correspondent actives and decoys molecules chosen from three databases (MUV, DUD-E and ChEMBL). With a similar motivation and trying to reproduce the impact that ImageNet (Deng et al., 2009) or WordNet (Fellbaum, 1998) had in computer science communities, MoleculeNet (Z. Wu et al., 2017) was designed. It is a benchmark collection for molecular ML containing data on the properties of nearly 600,000 compounds from different sources. This database is available in the previously mentioned DeepChem (Ramsundar, 2016b) package. In section 2.3, the MUV database (Rohrer and Baumann, 2009) was explained. This benchmark was specifically designed to be challenging for virtual screening analyses. The main objection to MUV is that only active molecules have been selected for avoiding biases, but no inactives, which could represent a bias by itself. It is a known problem that if the relative distances among actives properties are smaller than those between actives and inactives properties, machine learning algorithms can exploit these trivial similarities/differences to do the prediction (Wallach and Heifets, 2018).

Merck researchers claimed that **time-split validation** is more realistic than any random validation (Ma et al., 2015) and it is common practice in the pharmaceutical environment (Kearnes et al., 2017; Lenselink et al., 2017; Ramsundar et al., 2017; Sheridan, 2013). QSAR models are applied *prospectively*: first molecules assayed constitute the training set and those assayed later are the test set. The problem with this method of validation is that it requires the dates of testing, which are not available in most public domain datasets. Moreover, this kind of splitting has also shown to be biased, because of the same reasons exposed previously: active compounds discovered in the different phases would be very similar because of analogous chemical series in drug development (Kearnes et al., 2017; Wallach and Heifets, 2018).

Other techniques have been applied to reduce bias between training and validation sets. In Mayr et al., 2018 and Mayr et al., 2016, **compounds are clustered** based on their fingerprints distance prior to validation splitting. This ensures that all the data points from a cluster are either on the training or in the test set. In Xia et al., 2014, they did not only established chemical diversity of ligands, but also ensured that active molecules and their decoys were similar, so the classification model could not overfit based on physico-chemical differences. Similar approaches are taken in Wallach et al., 2015 (where again, unbiasing techniques only focus on redundancy between actives), Rodriguez-Perez and Bajorath, 2019 and Ramsundar et al., 2017.

Most methods mentioned above apply only to compounds. But in proteochemometrics models, the same bias problems can happen with **proteins**. Proteins can share high levels of similarity, both sequential and structural, especially when they belong to the same family. In Ragoza et al., 2017, they clustered data based on target families, ensuring that proteins with high percentage of sequence identity were retained in the same fold. In Karimi et al., 2019 they excluded some protein families from the training set to define the test set. This way, it could be tested if the model is able to generalize to unseen protein families. In Manica et al., 2019, they worked with cell lines instead of targets, making sure that no cell-drug pair in the test set had been seen before by the model. Similarly, Wan et al considered compound-protein pairs that appeared only once in the dataset as unique and used them as test set (Wan and Zeng, 2016).

2.5 OPEN ISSUES

As exposed throughout this chapter, the recent breakthrough of DL has promoted the adoption of a wide range of models in the fields of drug discovery and biological sequences analysis. Despite the rapid evolution of DL, there are still numerous limitations and unsolved problems. This section summarizes the open issues that will be specifically addressed in this thesis.

2.5.1 Lack of consensus on input representations

The first step when applying DL to a drug discovery problem is the decision of which input representation to use. There is no agreement on which is the best option and thus, the choice is usually taken based on a compromise between sparsity and concision, computational time and completeness of the study, manually engineered properties and automatic abstract representations, embedded and raw data, etc. (Qiu et al., 2017; van Westen et al., 2011). In proteochemometrics modelling, these decisions have to be extended to both compounds and targets. There are many studies comparing different types of input and claiming what they consider to be the optimal. However, the optimal is application-specific rather than universal. Many of the representations used lead to a loss of information that can affect the model performance.

In addition, machine learning implementations require all inputs to have the same dimensions. This poses a problem when using raw sequence data, given that every protein and every compound have a different composition length. Padding and truncation are possible solutions to this matter, but concrete steps or proper justification on the decision on how to handle it are usually neglected (Jurtz et al., 2017; Yu Li et al., 2018; Öztürk et al., 2018a; Rifaioglu et al., 2019). There is a lack of studies on the effect that this decision may have on the the models performance.

2.5.2 Influence of data bias in model performance estimates

Cheminformatics models have been shown to be little generalizable, collapsing when transferred to data from a different source (Altae-Tran et al., 2017; Ramsundar et al., 2015). The bias across different data sources could come from compounds of the same database sharing common structure or properties.

On the other hand, there is evidence that promising performance estimates of most reported ligand-based prediction models is due to overfitting to the training data rather than to good prospective accuracy (Kearnes et al., 2017; Wallach and Heifets, 2018). Applying a validation strategy that does not take into account similarity between molecules can lead to over-optimistic results (Wallach and Heifets, 2018). However, most of the studies still perform random cross-validation (Aliper et al., 2016; Dahl et al., 2014; Ghasemi et al., 2017; Goh et al., 2017c; Hamanaka et al., 2017; Jastrzebski et al., 2018; Korotcov et al., 2017; Koutsoukas et al., 2017; Kwon and Yoon, 2017; Öztürk et al., 2018a; Stepniewska-Dziubinska et al., 2018; Y. B. Wang et al., 2020; Y. Wang et al., 2017; Wen et al., 2017).

It is necessary to properly validate these models to avoid promoting novelty at the expense of accuracy (X. Yang et al., 2019). A robust strategy that considers data bias is needed to correctly evaluate cheminformatics models. Different approaches are being taken as time-split validation (Lenselink et al., 2017), compounds clustering (Rodriguez-Perez and Bajorath, 2019) or defining a common, robust benchmark for comparing results (Z. Wu et al., 2017).

2.5.3 Diagnosis and interpretability of CADD models

DL models have proven to be potentially useful regarding classical physics-based tools: they are computationally much cheaper and could help to accurately constrain the protein-ligand complexes to be simulated. DL tools are also more widely applicable than classical methods and do not require manual extraction of features (X. Yang et al., 2019). The performance of DL-based cheminformatics models is in general equivalent or superior to traditional ML models. However, in many cases this performance lift is not as significant when compared to the improvements that DNNs have brought to other fields such as speech recognition or computer vision (Goh et al., 2017b). In general, the incremental improvements in predictive accuracy are difficult to justify in the face of increase in model complexity (Goh et al., 2017b) and

computational cost (Ekins, 2016). All of this prevents the researchers external to the artificial intelligence community from embracing DL methods.

In order for the cheminformatics community to adopt such complex techniques, we should be able to understand their added value and to interpret how the protein and molecule spaces are being represented. But at the moment, DL methodologies are still seen as a black box by the life sciences researchers (Goh et al., 2017b). Thus, it is necessary to diagnose DL models in order to identify their strenghts and caveats. Although there are some advancements in this field (Manica et al., 2019; Shrikumar et al., 2017), there is still no standard way of diagnosing DL models.

2.5.4 Bioactivity data imbalance and data augmentation

Another central problem in the application of DL in CADD is the scarcity and quality of data, which despite the number of public molecular databases seen in section 2.3, is insufficient when compared to the data availability in other fields. Moreover, there are limited arrangements for open data-sharing in the pharmaceutical sciences (X. Yang et al., 2019). Increasing the size and quality of training data usually leads to a better generalization performance, but in drug discovery this is often costly or infeasible (X. Yang et al., 2019). A regularization technique known as *data augmentation*, which consists of artificially expanding a dataset in order to train the DL model on more examples, is highly used in other fields such as computer vision and speech recognition to compensate for the dataset size. However, this practice has not been widely adopted in DL-based CADD, which could be partially related to the lack of consensus in input representation (section 2.5.1).

An important issue in this regard is that bioactivity data is usually imbalanced. This imbalance can happens in both ways: on the one hand, databases gathering high-throughput screening results contain many more inactives than actives (Korkmaz, 2020) and on the other hand, databases built upon scientific literature have more actives than inactives because of the positive publication bias (Mlinarić et al., 2017). This imbalance can hinder the learning of the DL algorithms, since the quantity of data of the minority class is very limited (Korkmaz, 2020). Although the effect of data imbalance and the utility of resampling strategies in order to ease it have been tested in QSAR settings (Korkmaz, 2020; Zakharov et al., 2014), those have not been studied yet in proteochemometrics modeling.

3 GOALS

3.1 MAIN OBJECTIVE

DL models have shown to be very useful for target-ligand binding prediction within the context of sequential representations. However, many concerns have arised on the presence of bias in molecular databases, which may affect the performance of the models. Moreover, the lack of agreement on input representations, the disregard of some preprocessing steps and the bioactivity data imbalance could be interferring in the models behaviour. These issues are difficult to assess due to the lack of a standard way for interpreting, evaluating and understanding these models. The main objective of this thesis is the proper evaluation of DL-based models for binding affinity prediction of protein-molecule complexes based on informed decisions regarding input data preprocessing. Specifically, this thesis wants to establish a quality system for DL models in proteochemometrics applications.

3.2 DETAILED OBJECTIVES

The main objective of this thesis will be achieved through three conceptual blocks. In the first block, different validation strategies are assessed to address the data bias problem. In the second block, the effect of padding position in amino acid sequences is evaluated. In the third block, bioactivity data imbalance is analysed and its mitigation through oversampling techniques is investigated.

3.2.1 Improvement of validation strategies

- Identify the main source of bias in protein-compound binding affinity data.
- Assess the effect of the data bias in current validation strategies and quantify its impact on the performance of state of the art models.
- Propose a realistic validation schema, robust to data bias.

3.2.2 Study of the effect of padding sequences

- Define a family of padding strategies for biological sequences, including standard approaches and novel representations.
- Quantify the effect of padding position in the models performance in a biologically relevant classification problem.

- Understand the underlying differences between the padding strategies through the analysis of their lower dimensional representations.

3.2.3 Analysis of data balance

- Describe the protein-wise data imbalance in the context of a proteochemometrics classification problem.
- Pinpoint the main factor driving the proportion of active molecules predicted by the model.
- Investigate whether protein-wise data oversampling approaches have a positive impact in model performance.

3.3 EXPECTED CONTRIBUTIONS

This thesis is expected to bring two main contributions. First, a collection of recommendations on data preprocessing for DL-based target-compound models, providing sensible defaults for future applications. Second, a system for evaluating and diagnosing the design choices within the black box nature of DL models.

Every detailed objective is expected to lead to a publication in a indexed scientific journal. To encourage open and reproducible science, when our collaborators allow us to do so, the algorithms and models will be released as open source.

4 VALIDATION STRATEGIES

EVALUATION OF CROSS-VALIDATION STRATEGIES IN SEQUENCE-BASED BINDING PREDICTION USING DEEP LEARNING

Binding prediction between targets and drug-like compounds through Deep Neural Networks have generated promising results in recent years, outperforming traditional machine learning-based methods. However, the generalization capability of these classification models is still an issue to be addressed. In this work, we explored how different cross-validation strategies applied to data from different molecular databases affect to the performance of binding prediction proteochemometrics models. These strategies are: (1) random splitting, (2) splitting based on K-means clustering (both of actives and inactives), (3) splitting based on source database and (4) splitting based both in the clustering and in the source database. These schemas are applied to a Deep Learning proteochemometrics model and to a simple logistic regression model to be used as baseline. Additionally, two different ways of describing molecules in the model are tested: (1) by their SMILES and (2) by three fingerprints. The classification performance of our Deep Learning-based proteochemometrics model is comparable to the state of the art. Our results show that the lack of generalization of these models is due to a bias in public molecular databases and that a restrictive cross-validation schema based on compounds clustering leads to worse but more robust and credible results. Our results also show better performance when representing molecules by their fingerprints.

4.1 INTRODUCTION

Proteochemometrics or quantitative multi-structure-property-relationship modeling (QMSPR) is an extension from the traditional quantitative structure-activity relationship (QSAR) modeling (Andersson et al., 2011; Qiu et al., 2017; van Westen et al., 2011). In QSAR, the target protein is fixed and its interaction with ligands (small molecules or compounds) is predicted only from ligands descriptors. In contrast, the aim of proteochemometrics is to predict the binding affinity value by modeling the interaction of both proteins and ligands (Andersson et al., 2011). For this, a data matrix is built, each of its rows containing descriptors of both target and ligand linked to

This chapter is a postprint of the following journal article: Angela Lopez-del Rio, Alfons Nonell-Canals, David Vidal and Alexandre Perera-Lluna. "Evaluation of Cross-Validation Strategies in Sequence-Based Binding Prediction Using Deep Learning". *Journal of Chemical Information and Modeling, 59(4), 1645–1657 (2019).*

some experimentally measured biological activity. A statistical or machine learning method is then used to induce the model. The main advantages over QSAR are twofold: first, that the induced model can be applied for predictions of interaction with new proteins as well as ligands and second, that it can consider the underlying biological information carried by the protein as well as other possible cross-interactions of the ligand.

Deep learning (DL) is a branch of machine learning that stems from artificial neural networks, which are computational models inspired in the structure of the brain and the interconnection between the neurons. DL is able to learn representations of raw data with multiple levels of abstraction (LeCun et al., 2015). These concepts started to be developed in the 1940s (McCulloch and Pitts, 1943) but it was not until 2012 that there was a break through of the Deep Neural Networks (DNN) (Krizhevsky et al., 2012). Since then, DL has been successfully applied in natural language processing (Young et al., 2018), image recognition (K. He et al., 2016), drug discovery (Gawehn et al., 2016) or computational biology (Angermueller et al., 2016). The increase of computational power by parallel computing with graphics processing units (GPU) and the improvement of optimizers (G. Hinton et al., 2012; Diederik P Kingma and Ba, 2014) and regularization techniques (Ioffe and Szegedy, 2015; N. Srivastava et al., 2014) contributed to this resurgence, along with the development of software platforms that allow to make prototyping faster and automatically manage GPU computing, like Theano (Theano Development Team, 2016) or Tensorflow (Martín Abadi et al., 2015).

DL provides a framework for the identification of both of biological targets and biologically active compounds with desired pharmacological effects (Gawehn et al., 2016). In 2012, DNN won a QSAR machine learning challenge on drug discovery and activity prediction launched by Merck (*Merck Activity Competition* 2012), outperforming Merck's Random Forests baseline model by 14% accuracy (Dahl et al., 2014). Since then, the application of DL to pharmaceutical problems gained popularity (Gomes et al., 2017; Kadurin et al., 2017a; Kearnes et al., 2017; Kwon and Yoon, 2017; Mayr et al., 2016, 2018; Neil et al., 2018; Ramsundar et al., 2015; K. Tian et al., 2016; Unterthiner et al., 2014; Wallach et al., 2015), although it has been mainly applied to multitask QSAR modeling. Regarding DL-based proteochemometrics, little has been done except for the work of Lenselink et al (Lenselink et al., 2017), where they compared different machine learning methods for proteochemometrics, DNN being the top performer.

Independently of the machine learning technique used, a curated design of the cross-validation strategy is critical for the proper evaluation of the binding prediction model. The predictive power of a consistent model must remain stable when applied to data that comes from a different source than the training set. Moreover, possible redundancy in the data must be controlled. Proteins are divided into families, which usually have similarities in sequence or structure. Compounds might be part of the same chemical series. The performance of classification model should be tested when applied to families of proteins or compounds with different scaffolds than those used to train it. On the latter, Wallach and Heifets concluded that performance of most of the reported ligand-based classification problems reflect overfitting to training benchmarks rather than good prospective accuracy (Wallach

and Heifets, 2018), mainly because of the redundancy between training and validation sets. This issue becomes more critical when using random cross-validation: in the pharmaceutical field, compounds are usually synthesized serially to enhance molecular properties. This leads to training and validation sets following the same distribution, which is desirable in most machine learning problems, but a poor estimate of reality in drug discovery (Kearnes et al., 2017).

Time-split validation is common practice in pharmaceutical environment to overcome this issue (Kearnes et al., 2017; Lenselink et al., 2017; Ma et al., 2015; Sheridan, 2013). This strategy is well suited to the realistic scenario, where we are interested in prospective performance of the models (Sheridan, 2013). However, time-split data has also shown to be biased because of the high similarity between actives discovered in different phases (Kearnes et al., 2017; Wallach and Heifets, 2018), so this cross-validation strategy would not be suited for generalization across chemical classes.

Other techniques have been applied to reduce bias and data redundancy between training and validation sets. Unterthiner et al. (Mayr et al., 2018; Unterthiner et al., 2014) clustered compounds using single linkage to avoid having compounds sharing scaffolds across training and validation sets. Rohrer and Baumann designed the Maximally Unbiased Validation (MUV) benchmark to be challenging for standard virtual screening: actives have been selected to avoid biases of enrichment assessment and inactives have been biologically tested against their target (Rohrer and Baumann, 2009). Xia et al. (Xia et al., 2014) presented a method to ensure chemical diversity of ligands while keeping the physicochemical similarity between ligands and decoys. Wallach et al. removed analogue bias in active molecules by clustering and selected decoys to match in sets to actives with respect to some 1D physicochemical descriptors while being topologically dissimilar based on 2D fingerprints (Wallach et al., 2015). However, these unbiasing techniques only focus on redundancy between actives, overlooking the impact of inactive-active or inactive-inactive similarity, which leads to models memorizing the similarity between benchmark inactives and hence, overfitting (Wallach and Heifets, 2018).

Another related issue is that the possible bias across the different data sources used in some studies has not been properly studied yet (Wallach et al., 2015). Different datasets might have different structure, affecting to the study of the generalization of the model. A related issue is found in the study of Altae-Tran et al (Altae-Tran et al., 2017), where after the collapse of their transfer learning experiments it is affirmed that one-shot learning methods may struggle to generalize to novel molecular scaffolds, and that there is a limit to their cross-tasks generalization capability.

Analysis of bias in binding classification models have been always focused on QSAR models, but how the inclusion of protein information could bias the QMSPR model remains unknown. Proteins are macromolecules constituted by amino acid residues covalently attached to one another, forming long linear sequences which identify them, defining its folding and its activity. The main value of DL in this context is that DL can directly learn from the sequence, capturing nonlinear dependencies and interaction effects, and

hence providing additional understanding about the structure of the biological data.

The appropriate DL architecture to manage this kind of data are bi-directional Recurrent Neural Networks (RNN), well suited for modeling data with a sequential but non-causal structure, variable length and long-range dependencies (Lipton et al., 2015). Baldi et al have applied bi-directional RNN to protein sequence for predicting secondary structure (Agathocleous et al., 2010; Baldi et al., 1999; Baldi and Pollastri, 2003), for matching protein beta-sheet partners (Baldi et al., 2000) or for predicting residue-residue contact (Di Lena et al., 2012). However, classical RNN cannot hold very long-range dependencies and to overcome this issue Hochreiter et al applied Long Short Term Memory (LSTM) networks to classify amino acid sequences into superfamilies (Hochreiter et al., 2007). Jurtz et al applied bi-directional LSTM to amino acid sequence for subcellular localization, secondary structure prediction and peptides binding to a major histocompatibility complex (Jurtz et al., 2017).

Additionally, another DL architecture is specially useful when working with biological sequences: Convolutional Neural Networks (CNN). Convolutional layers imply translational invariance (Kauderer-Abrams, 2017), so it is not needed for sequences to be aligned (not even alignable, consequently). This is an advantage over traditional sequence-based target descriptors which require multiple target alignment (Qiu et al., 2017). On the other hand, these layers are useful for detecting patterns of pre-defined lengths with biological relevance.

In this paper, we analyze and quantify the effect of different cross-validation strategies on the performance of binding prediction DL-based proteochemometrics models. In order to evaluate this problem in a exhaustive way, we compare these DL models with baseline logistic regression (LR) models and explore different representations for molecules.

4.2 MATERIALS AND METHODS

4.2.1 Data

Original database	# targets	# compounds	# compounds/target mean (SD)	% actives mean (SD)
ChEMBL	50	29,986	599.7 (0.9)	16.7 % (0.1)
DUD	21	12,417	591.3 (80.5)	14.2 % (9.4)
MUV	17	9,001	529.5 (1.1)	5.7 % (0.0)
Total	**88**	**51,404**	**584.1 (47.2)**	**13.9 % (6.1)**
Unique	83	32,950	-	-

Table 4: Number of targets, number of compounds, average number of compounds per target and average percentage of actives for each source database present in the Riniker et al dataset (Riniker and G. A. Landrum, 2013) after adapting it for our study. SD: standard deviation.

Models were trained on the dataset generated from three different publicly available sources by Riniker and Landrum (Riniker and G. A. Landrum,

2013). The aims of this dataset are true reproducibility and comparability of benchmarking studies. This dataset incorporates 88 targets from ChEMBL (Bento et al., 2014), the Directory of Useful Decoys (DUD) (N. Huang et al., 2006) and the MUV (Rohrer and Baumann, 2009). The sequence similarity between the selected targets can be seen in Figure S40 of the Supporting Information. The selection of ChEMBL targets was based on the 50 human targets and actives proposed by Heikamp and Bajorath study (Heikamp and Bajorath, 2011) and performed on ChEMBL version 14 (Riniker and G. A. Landrum, 2013). DUD targets with more than 30 available actives were used. All targets from MUV were used. The distribution of protein families among the selected targets can be seen in Figure S41 of the Supporting Information.

In the Riniker and Landrum dataset the selection of actives and decoys was conducted on drug-like molecules and in such a way as to cover the maximum range of the chemical spectrum, based on diversity and physical properties. In the case of ChEMBL, decoys were selected from the ZINC database (Irwin et al., 2012). For MUV and DUD, decoys came from the original database. As a result, in the Riniker and Landrum database the number of decoys per target was variable, ranging from 1,344 to 15,560. We randomly selected 500 molecules from each original set of decoys in order to have a more computationally-approachable dataset and to decrease active/decoy imbalance per target while keeping a plausible proportion. The list of molecules (both active and inactive) identified by their SMILES (Weininger, 1988) was then standardized to avoid multiple tautomeric forms. Finally, these compounds were filtered to remove salts, those with molecular weight >900Da or >32 rotatable bonds and those containing elements other than C, H, O, N, S, P or halides. Table 4 provides a summarized description of the final dataset used in our study, while Table S13 of the Supporting Information contains more detailed information for each target of the dataset (ID and description, source database and number of actives/decoys).

4.2.2 Prospective test

We used an external test set in order to obtain an alternative estimation of the performances of the models using a prospective design. This test set was built from the data added in ChEMBL version 23 (May 2017) with respect to ChEMBL version 14 for the 88 targets of Riniker and Landrum dataset, with the aim of setting up a prospective prediction set. Only valid, conclusive and human (or from homologous organisms, H) assays of direct interactions (D) posterior to 2013 (year of release of ChEMBL 14) were extracted. Activity values were log-transformed and thresholded at 6 to define active/inactive labels. As a result, we obtained 3,118 unseen active and 787 inactive pairs from ChEMBL 23.

In order to keep the active/inactive proportion of the original sets, decoys from ZINC15 were added. For each molecule with an assay labeled as active, a disjunct pool of 255 random molecules from ZINC15 was built. Two decoys with a Dice similarity > 0.5 computed from an atom-count fingerprint (ECFC0) were chosen from each pool, following the decoys generation procedure of Riniker and Landrum (Riniker and G. A. Landrum, 2013).

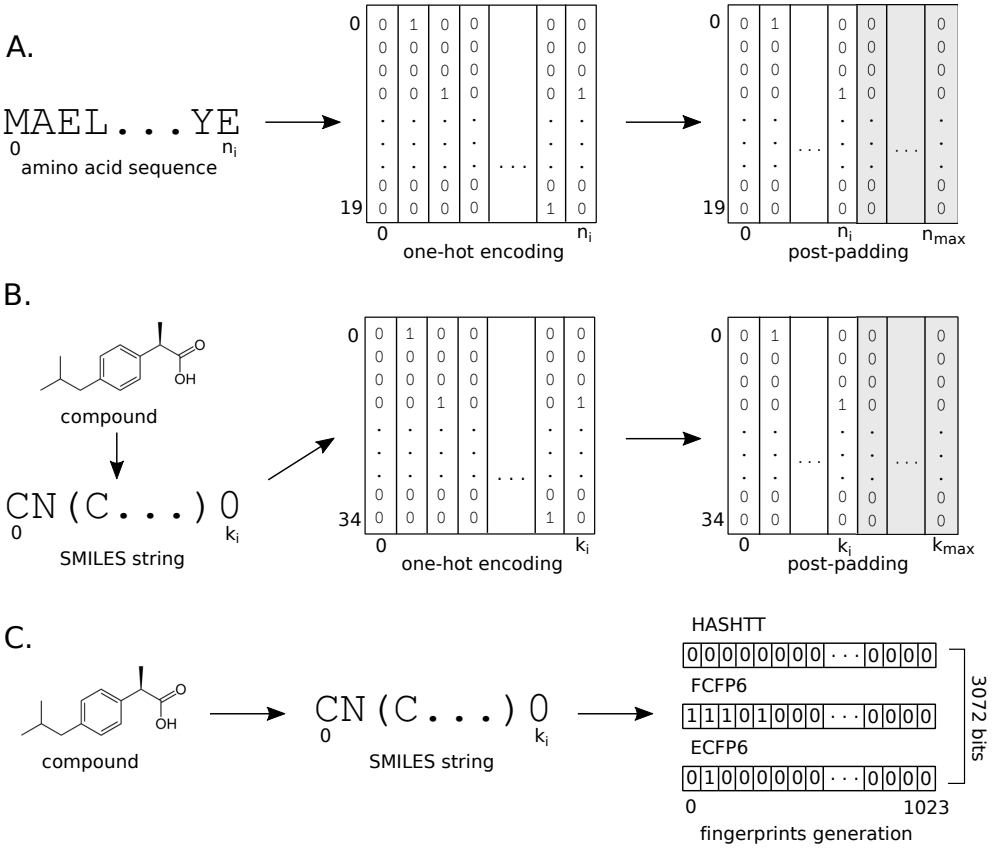

Figure 19: Schema of descriptors and encoding process of the different inputs of the model A. Amino acid sequence is first one-hot encoded and then padded at the end to the length of the longest target (n_{max}), in this case 1987. **B.** Compound represented by its SMILES identifier is also one-hot encoded and then padded to the length of the longest SMILES string (k_{max}), in this case 93. **C.** Compound described by its fingerprints is first identified by its SMILES, from which HASHTT, FCFP6 and ECFP6 bit strings are generated.

All the molecules followed the same standardization and filtering specified in the Data subsection. It was assured that none of the molecules of the external common test set was also present on the original Riniker and Landrum dataset. The final external common test set is composed of 8,446 pairs of target-compound (5,856 inactive and 2,590 active from 7,991 unique compounds) from 22 of the targets from the original dataset. A more detailed description of the dataset per target is shown in Table S15 of Supporting Information.

4.2.3 Descriptors

We tested two ways of representing input molecules: (1) as sequences of symbols, using the SMILES notation and (2) as the combination of molecular fingerprints (Todeschini and Consonni, 2008), where structural information is represented by bits in a bit string. The SMILES representation as input for a DL model was based on the DeepCCI by Kwon et al (Kwon and Yoon,

2017). Model input has to be numerical, so SMILES notation was one-hot encoded (Figure 19B). This means that every character of the SMILES string was represented by a binary vector of size 35, with all but its corresponding entry set to zero. SMILES were padded to the length of the longest string, 94.

For the fingerprints representation we selected three of them: topological torsions (TT) fingerprint (Nilakantan et al., 1987), extended connectivity fingeprint and functional connectivity fingerprint, both with a diameter of 6 (ECFP6 and FCFP6, respectively) (Rogers and Hahn, 2010) (Figure 19C). TT describe four atoms forming a torsion, and the atom type includes the element, the number of non-hydrogen neighbors and number of π electrons. ECFP6 and FCFP6 encode circular atom environment up to radius 3. In ECFP6, atom type includes the element, the number of heavy-atom neighbors, the number of hydrogens, the isotope and ring information. FCFP6 use pharmacophoric features. All of them were generated using the RDKit package (G. Landrum, n.d.), and defined with a length of 1024 bits, since there is proof of a very low number of collisions with this size (Riniker and G. A. Landrum, 2013).

For protein representation, raw amino acid sequences were fed to the model (Figure 19A). As for SMILES strings, these sequences were converted to numerical through one-hot encoding, only that in this case each amino acid was represented by a binary vector of length 20. Amino acid sequences were then padded to the length of the longest target, in this case, 1988. Due to the translational invariance of the convolutional layers of the model, neither amino acid nor the SMILES sequences needed to be aligned.

4.2.4 Cross-validation strategies

Four different cross-validation strategies were applied to both active and inactive Riniker dataset compounds (see Figure 20A), omitting binding targets. Potential applications of each of these strategies are discussed in Table S14 of the Supporting Information. In all cases active/inactive proportion was preserved for training, validation and test sets. (1) **Random**, where compounds were randomly split 80/10/10 in training, validation and test with no further criteria (Figure 20B). (2) **Database-based**, where division in training, validation and test was performed according to the source database of the compounds (ChEMBL, MUV and DUD, as seen in Data section) (Figure 20C). (3) **Clustering-based**, where K-Means clustering with k=100 was applied to the fingerprint description of molecules (see Figure S42 of the Supporting Information). This was used to avoid having similar molecules both in training and validation/test set and thus control for the compound series bias (Mayr et al., 2018; Wallach and Heifets, 2018). Each cluster was randomly assigned to one of the split sets until achieving 80/10/10 splitting (Figure 20D). This strategy is similar to the leave-class-out method (Lombardo et al., 2004). (4) **Intermediate**, where the previous K-Means clustering was also applied, but only to those compounds coming from ChEMBL (Figure 20D). In order for this schema to have a test set of comparable size with the others, only one data source was used. We chose MUV dataset since

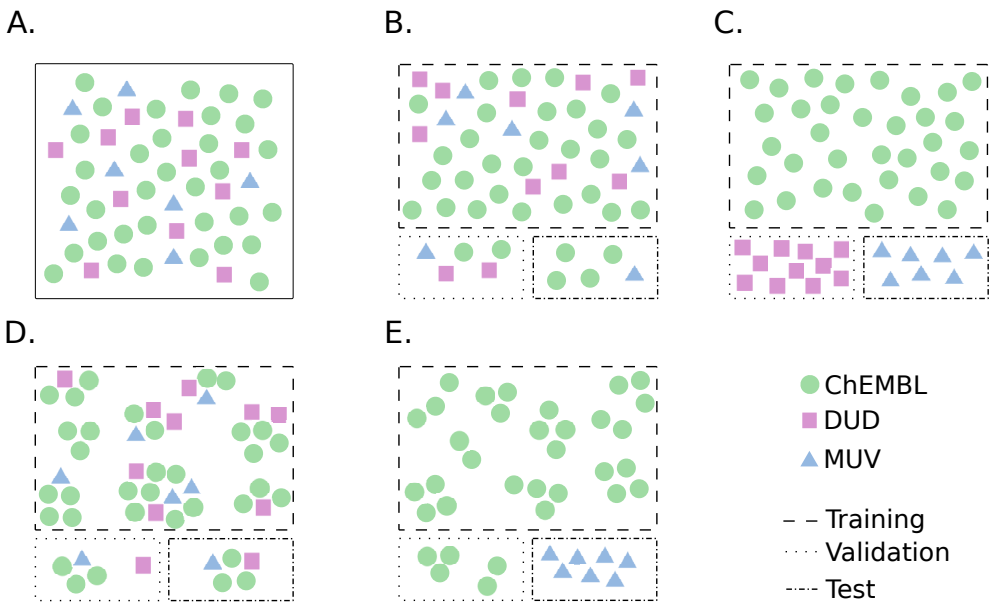

Figure 20: **Different cross-validation strategies** applied to the Riniker et al dataset (Riniker and G. A. Landrum, 2013). **A.** Original dataset **B.** Random splitting of the compounds. **C.** Database-based division of compounds. **D.** Clustering-based splitting. **E.** Intermediate splitting.

it was designed to be challenging, as seen in the Introduction, while data architectural design of the original DUD is not that well suited for this problem (Mysinger et al., 2012). In Figures S43-S46 of the Supporting Information, the proportion of actives/inactives for each target in each splitting set is depicted for random, clustering, database-based and intermediate cross-validation strategies, respectively. In all cases, the training set was used to fit the model, the validation set was used to evaluate the model fit in each epoch and tune hyperparameters accordingly and the test set was used to externally evaluate the final model fit.

4.2.5 Prediction models

A DL-based model was built to evaluate the effect of different cross-validation strategies on binding prediction. A LR model was also trained to have a baseline to compare with. Besides, two different molecular representations were tested to evaluate their performance: their SMILES string and three selected fingerprints (ECFP6, FCFP6 and HASHTT). Each one of these models was trained according to the four cross-validation strategies presented in the previous section. As a result, a total of 16 different models were evaluated.

Logistic Regression

This baseline model consisted on two input layers concatenated, one for the compound and one for the target, with as many neurons as the size of the input (94 and 3072 in the case of SMILES and fingerprints, respectively and 1988 in the case of targets) connected to a sigmoidal unit (see Figure S47

of the Supporting Information). It is expected for this baseline model not to be optimal to process amino acid sequences or SMILES strings.

Deep Learning models

An schematic representation of the DL predictive models used can be seen in Figure S48 of the Supporting Information, A and B. The amino acid sequence analysis block is common for both models and it is a Convolutional Recurrent Neural Network based on the one used by Jurtz et al for the prediction of subcellular localization of proteins (Jurtz et al., 2017) (Figure S48C). This architecture allows to build complex representations from both targets and compounds for the prediction of binding, in contrast to the LR models, which are directly fed with the input features.

The input of the model is the amino acid sequence one-hot encoded, which is passed through a 1D CNN (Krizhevsky et al., 2012). This 1D CNN comprises filters of sizes 1, 3, 5, 9, 15 and 21, with the aim of detecting motifs of different length in the amino acid sequence. Convolutional layers are followed then by a max pooling layer. The input is then introduced to a bi-LSTM neural network (Lipton et al., 2015). Dropout is used to prevent overfitting (N. Srivastava et al., 2014).

The compound analysis block depends on the encoding of molecules. If molecules are represented by their SMILES, then the compound processing block is similar to the sequence processing block (Figure S48A). The SMILES string is one-hot encoded and the input is passed to a bank of convolutional filters of size 3, 4 and 5 based on the LINGO substrings analysed by Vidal et al (Vidal et al., 2005). After that, a maximum pooling layer transfers the information to a LSTM. The LSTM is uni-directional since the SMILES strings are causal, in the sense that they are read in only one direction. Dropout is also used here to prevent overfitting. On the other hand, when molecules are represented by ECFP6, FCFP6 and HASHTT fingerprints, namely a binary vector of length 3072, the input is passed through a feed-forward neural network (Ramsundar et al., 2015; Unterthiner et al., 2014) followed by dropout (Figure S48E).

Finally, the sequence and the compound analysis blocks are merged and the information is processed by a softmax activation unit, which quantifies how likely the sequence-compound binding is. Binary predictions are obtained by thresholding the activation at 0.5.

4.2.6 Implementation

Both models (DL and LR) were trained with Adam optimizer (Diederik P Kingma and Ba, 2014) for 500 epochs, with a batch size of 128 for training and 64 for validation (learning rate=5e-6 for DL models encoded by fingerprints, 5e-5 for the rest). Decay rate was defined as learning rate/number of epochs. The other parameters were set as proposed by Jurtz et al (β_1=0.1, β_2=0.001, ϵ=1e-8) (Jurtz et al., 2017).

Models were implemented in Python (Keras (Chollet et al., 2015) > 2.0 using as backend TensorFlow (Martín Abadi et al., 2015) > 1.4) and run on the GPU NVIDIA TITAN Xp and NVIDIA GeForce GTX 1070.

4.2.7 Characterization of data structure

Each one of the cross-validation strategies is analyzed in terms of imbalance and data redundancy to better understand and interpret performance results. First, active/inactive proportion is explored for each cross-validation schema. Then, overlap of targets and compounds between split sets is computed as a percentage with respect to the total number of targets (88) and compounds (32,950), respectively. After that, Tanimoto similarity (B. Zhang et al., 2015) based on FCFP6 fingerprint was calculated between split sets for each cross-validation strategy. It was computed between each pair of compounds (in terms of compound-target pair) both inter- and intra-split sets. An analysis of the number of similar compounds in the test set for each compound in the training set was performed to assess model generalization (similarity threshold = 0.7). Lastly, distribution of chemotypes and protein classes is explored for each strategy. For targets, this distribution is studied for the main protein families. Since for molecules there is no such classification, we decided to group them in terms of their Bemis-Murcko scaffold (BMS) (Bemis and Murcko, 1996), a technique for extracting molecular frameworks by removing side chain atoms which has been used for clustering compounds (Gomes et al., 2017; Mysinger et al., 2012; Wallach et al., 2015).

4.2.8 Performance metrics

Area under the receiver operating characteristic (ROC) curve (from now, referred to as AUC), traditionally employed for measuring the performance of classification models, has been reported for not being enough for evaluating virtual screening models because it is not sensitive to early recognition and it is affected by class imbalance (Truchon et al., 2007). Thus, we complement this information with partial AUC (pAUC) at 5%, which allows to focus on the region of the ROC curve more relevant for virtual screening (Andersson et al., 2011) (up to 5% of the False Discovery Rate), with Cohen's kappa coefficient (κ) (Cohen, 1960), which measures the agreement between real and predicted classification, and with the Boltzmann-enhanced discrimination of receiver operating characteristic (BEDROC), a metric proposed to overcome the limitations of AUC (Truchon et al., 2007) increasingly popular in the evaluation of virtual screening models. BEDROC uses an exponential function based on parameter α and is bounded between 0 and 1, making it suitable for early recognition. As recommended by Riniker et al (Riniker and G. A. Landrum, 2013), we focus on AUC and BEDROC with α=20, whilst also reporting α=100.

We implemented and trained DL and LR binding classification models. We then selected the best training epoch in terms of F1 Score, the harmonic mean of precision and recall, on the validation set, since it can handle class imbalance and encourages the prediction of both actives and inactives. Finally, we tested the selected models on the corresponding test set of each cross-validation strategy. AUC, BEDROC and pAUC were computed from raw predicted probabilities, while κ and F1 Score were calculated from the binarized predictions. Stratified subsampling of the 80% of the test data was

used to sample 100 values from the performance estimates distributions (subsampling estimates). Models performance was also analyzed per target and by target family. The nonparametric Wilcoxon rank-sum test (Hollander and Wolfe, 1999) was used to compare AUC and BEDROC(20) metrics between all pairs of models. P-values were adjusted for multiple testing by computing the False Discovery Rate (FDR) by Benjamini-Hochberg (Benjamini and Hochberg, 1995) for each metric.

This study explores multiple factors (i.e. validation type, cross-validation strategy, encoding, algorithm and protein family), which makes direct comparisons cumbersome. To overcome this issue, the performance metrics were described through linear models built upon those factors. Such explanatory models, which have been previously used in other scenarios for similar purposes (Picart-Armada et al., 2019), allow for a formal, quantitative statistical comparison between the factors.

Differences in performance between the four validation strategies were described and tested in terms of two explanatory linear models.

$$\text{auc/bedroc}(20) \sim \text{cv} + \text{encoding} + \text{algorithm} \tag{7}$$

Per-target analysis was handled with the following additive model, whose `family` covariate took into account the target type:

$$\text{bedroc}(20) \sim \text{cv} + \text{encoding} + \text{algorithm} + \text{family} \tag{8}$$

Changes in performance estimates when switching to the prospective validation were quantified in the model below, which included an interaction term (represented by ":") with the type of validation `valtype` (either cross-validation or prospective).

$$\text{auc/bedroc}(20) \sim \text{cv} + \text{encoding} + \text{algorithm} + \text{valtype} : \text{cv} \tag{9}$$

Reference categories were: random `cv`, LR `algorithm`, fingerprints for `encoding`, CY (Cytochromes P450) for `family` and cross-validation for `valtype`. All the models were fitted on the estimated performances, rather than the subsampling estimates. The conventional model parametrizations in the R statistical programming language (R Core Team, 2015) were followed. Each term represents the difference between its specified category and the reference category of that variable. The constant term of a model reflects its prediction when all the categorical variables are set to their default categories.

4.3 RESULTS

4.3.1 Cross-validation strategies analysis

In Figure 21 and S50 of the Supporting Information, active/inactive imbalance in each split set is shown for each cross-validation strategy both in absolute and relative terms, respectively. Active/inactive proportion maintains for all sets.

Figure 22A shows overlap of targets between split sets for each cross-validation schema. In clustering-based and random, every target is repeated

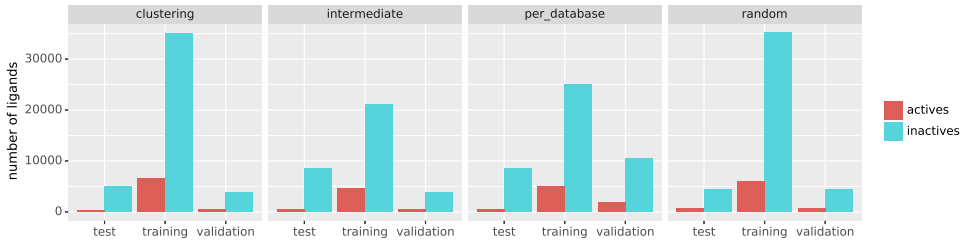

Figure 21: Proportion of active/inactive compounds in each set (training, validation and test), for each cross-validation strategy.

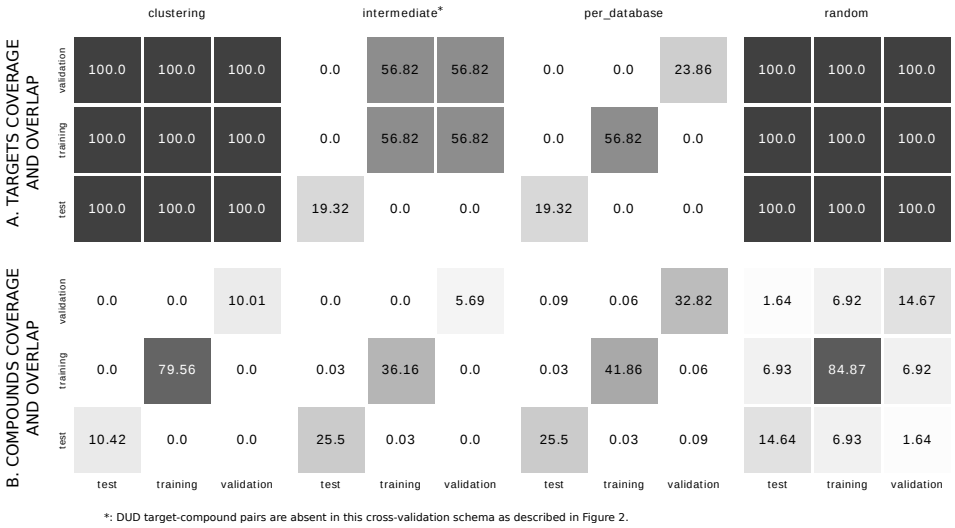

Figure 22: **A.** Coverage and overlap of targets between splitting sets. Numbers inside tiles refer to the percentage of overlapping targets respect to the total number of targets, 88. **B.** Coverage and overlap of compounds between splitting sets. Numbers inside tiles refer to the percentage of overlapping compounds respect to the total number of compounds, 32,950.

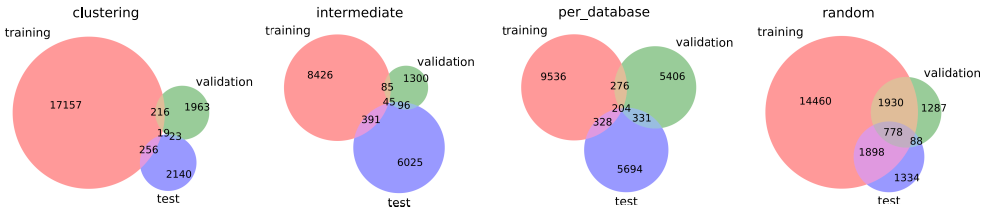

Figure 23: Overlap of BMS between split sets for each cross-validation strategy.

on the three splitting sets. In intermediate we only find overlapping targets between training and validation, since both splitting sets are built from the ChEMBL dataset, while test corresponds to the MUV database. In database-based, there is no overlap of targets between splitting sets.

Figure 22B shows overlap of compounds between splitting sets for each cross-validation schema. In this case, in clustering-based there are no repeated compounds across groups. In database-based and intermediate there is negligible overlap between groups, probably due to repeated inactives for different targets. Since in random cross-validation splitting was made randomly, there are repeated compounds in all splitting sets. In Figure S51 of the Supporting Information the same representation can be seen only for active molecules, following the same behavior but with a smaller magnitude.

Table 5 quantifies the similarity from each molecule of the test set for all the molecules in the training set in each cross-validation strategy. The percentage of similar compounds is one order of magnitude higher in random cross-validation than in the rest of schemes. A histogram on their distribution can be found in Figure S52 of the Supporting Information. Figure S53 of the Supporting Information shows Tanimoto similarity intra- and inter- split sets for each cross-validation strategy.

Table 5: Number and proportion of similar compounds in the training set for each molecule from the test set for each cross-validation strategy in terms of protein-compound pairs. % similarity ± confidence interval 95% (computed based on the formula in Daniel, 1999). Similarity threshold = 0.7. CV: cross-validation.

CV strategy	molecules in test	total similar instances in training	molecules in test with ⩾ 1 similar instances in training	% similarity
clustering	5,301	518	236	4.45% ± 0.56
intermediate	9,001	329	127	1.41% ± 0.24
per_database	9,001	478	183	2.03% ± 0.29
random	5,141	8,203	3,273	63.66% ± 1.31

In Figure 23, Bemis-Murcko molecular scaffolds overlap between split sets is shown. As above, in random cross-validation there are more overlapping scaffolds between training, test and validation. In the other strategies the number of overlapping scaffolds decreases significantly. Training set in clustering-based cross-validation has the highest number of different scaffolds.

In Figure 24 distribution of protein families between split sets for each validation strategy is represented. G protein coupled-receptors and protein kinases are the most numerous families in training sets. Every group has targets of each protein family, except for the validation set in the database-

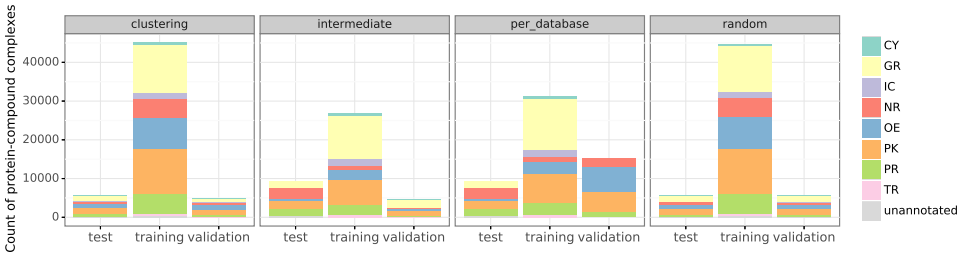

Figure 24: Distribution of protein families between split sets for each cross-validation strategy. Y-axis shows the count of all active/inactive protein-compound pairs. CY: cytochromes P450, GR: G protein-coupled receptors, IC: ion channels, NR: nuclear receptors, OE: other enzymes, PK: protein kinases, PR: proteases, TR: transporters.

based schema which lacks some protein families (transporters, cytochromes, G protein-coupled receptors and ion channels).

4.3.2 Models performance

	AUC	BEDROC(20)
cv clustering	-0.087^{***} (-0.133, -0.041)	-0.326^{***} (-0.417, -0.235)
cv perdb	-0.340^{***} (-0.386, -0.293)	-0.726^{***} (-0.817, -0.634)
cv intermediate	-0.345^{***} (-0.391, -0.299)	-0.721^{***} (-0.813, -0.630)
encoding smiles	-0.066^{***} (-0.098, -0.033)	-0.090^{**} (-0.154, -0.025)
algorithm dl	0.033^{*} (0.0004, 0.066)	0.052 (-0.012, 0.117)
Constant	0.891^{***} (0.851, 0.931)	0.850^{***} (0.771, 0.930)
Observations	16	16
R^2	0.972	0.972
Adjusted R^2	0.959	0.958
Residual Std. Error (df = 10)	0.033	0.066
F Statistic (df = 5; 10)	70.529^{***}	69.622^{***}
Note:		*p<0.1; **p<0.05; ***p<0.01

Table 6: Linear models on AUC and BEDROC(20). The reference levels were omitted.

A summary of performance metrics of each model on its test set can be seen on Table 7, whereas the models in Equation 7 that describe the performance estimates are summarized in Table 6.

Models based on the random cross-validation schema yielded the highest performance estimates for both DL and baseline LR algorithms. In turn, the performance estimated through clustering was higher than that of database and intermediate strategies, which were indistinguishable. All pairwise comparisons except database/intermediate were significant at $p < 0.05$ after adjusting for multiple testing, Tukey's method, see Tables S16 and S17. The architecture based on the fingerprints representation of compounds had a better performance than the SMILES encoding, ($p < 0.05$, see Table 6). Regarding the algorithm, results are not conclusive on which one performs better ($p > 0.05$ in Table 6).

CV strategy	Encoding	Algorithm	Best epoch	F1 score	AUC	pAUC(5%)	BEDROC(20)	BEDROC(100)	κ
Random	Fingerprints	LR	496	0.722 ± 0.006	0.910 ± 0.003	0.029 ± 0.001	0.868 ± 0.007	0.940 ± 0.022	0.678 ± 0.007
		DL	392	$\mathbf{0.859 \pm 0.005}$	$\mathbf{0.961 \pm 0.002}$	$\mathbf{0.040 \pm 3.8 \cdot 10^{-4}}$	$\mathbf{0.956 \pm 0.004}$	$\mathbf{0.981 \pm 0.012}$	$\mathbf{0.835 \pm 0.006}$
	SMILES	LR	499	0.534 ± 0.009	0.807 ± 0.006	$0.018 \pm 4.7 \cdot 10^{-4}$	0.741 ± 0.009	0.973 ± 0.004	0.465 ± 0.009
		DL	498	0.568 ± 0.008	0.824 ± 0.005	$0.020 \pm 4.5 \cdot 10^{-4}$	0.763 ± 0.008	0.961 ± 0.005	0.506 ± 0.009
Clustering	Fingerprints	LR	491	0.409 ± 0.011	0.834 ± 0.007	0.015 ± 0.001	0.529 ± 0.012	0.532 ± 0.034	0.358 ± 0.011
		DL	402	$\mathbf{0.501 \pm 0.010}$	$\mathbf{0.852 \pm 0.007}$	$\mathbf{0.024 \pm 0.001}$	$\mathbf{0.684 \pm 0.012}$	$\mathbf{0.779 \pm 0.029}$	$\mathbf{0.460 \pm 0.011}$
	SMILES	LR	489	0.322 ± 0.011	0.727 ± 0.008	0.012 ± 0.001	0.414 ± 0.012	0.519 ± 0.022	0.277 ± 0.011
		DL	495	0.294 ± 0.012	0.736 ± 0.008	0.011 ± 0.001	0.396 ± 0.013	0.569 ± 0.022	0.258 ± 0.013
Database based	Fingerprints	LR	22	0.033 ± 0.004	0.536 ± 0.007	$0.001 \pm 1.2 \cdot 10^{-4}$	0.095 ± 0.005	0.041 ± 0.005	0.001 ± 0.004
		DL	20	$\mathbf{0.094 \pm 0.006}$	$\mathbf{0.556 \pm 0.007}$	$\mathbf{0.002 \pm 1.8 \cdot 10^{-4}}$	0.115 ± 0.007	0.082 ± 0.011	0.030 ± 0.006
	SMILES	LR	230	0.013 ± 0.004	0.497 ± 0.007	$0.001 \pm 1.5 \cdot 10^{-4}$	0.081 ± 0.006	0.056 ± 0.009	$2.8 \cdot 10^{-4} \pm 0.004$
		DL	492	0.075 ± 0.007	0.551 ± 0.007	$\mathbf{0.002 \pm 2.1 \cdot 10^{-4}}$	$\mathbf{0.133 \pm 0.007}$	$\mathbf{0.112 \pm 0.012}$	$\mathbf{0.034 \pm 0.007}$
Intermediate	Fingerprints	LR	57	0.021 ± 0.004	0.518 ± 0.006	$0.001 \pm 1.4 \cdot 10^{-4}$	0.088 ± 0.005	0.050 ± 0.008	-0.003 ± 0.004
		DL	268	$\mathbf{0.123 \pm 0.005}$	$\mathbf{0.553 \pm 0.007}$	$0.002 \pm 1.5 \cdot 10^{-4}$	0.132 ± 0.007	0.105 ± 0.017	$\mathbf{0.037 \pm 0.005}$
	SMILES	LR	428	0.014 ± 0.003	0.495 ± 0.010	$0.001 \pm 1.4 \cdot 10^{-4}$	0.082 ± 0.006	0.060 ± 0.009	$-3.8 \cdot 10^{-5} \pm 0.003$
		DL	475	0.064 ± 0.006	$\mathbf{0.553 \pm 0.007}$	$\mathbf{0.003 \pm 2.0 \cdot 10^{-4}}$	$\mathbf{0.139 \pm 0.007}$	$\mathbf{0.120 \pm 0.012}$	$\mathbf{0.037 \pm 0.006}$

Table 7: Summary of performance on the test set of the different cross-validation strategies for DL and LR models, in terms of F1 score, AUC, partial AUC (pAUC) at 5%, BEDROC(α=20) and BEDROC(α=100). Mean $\pm$ standard deviation of the subsampling estimates. In bold, the best performance for each CV strategy. CV: cross-validation.

In general, the best algorithm depends of the cross-validation strategy and the compound representation, but there is a tendency of logistic regression being better for the SMILES encoding, and deep learning for the fingerprints representation, see Figure 25.

In Figure 25, the AUC and BEDROC(20) of the cross-validation strategies are compared between them and with a random prediction. The order of the performance estimates of the strategies from Table 6 can be appreciated. Database-based and intermediate strategies have both poor performance, comparable to that of a random prediction. The same behavior can be seen in ROC curves of all possibilities in Figure S54 of the Supporting Information.

The mean difference of BEDROC(20) metrics for each possible combination of models is shown in Figure 26. Differences in terms of AUROC follow the same behavior but of a smaller magnitude (see Figure S55 of Supporting Information). The most remarkable differences can be seen on random models versus database-based and intermediate models. There are also relevant differences on performance between clustering-based cross-validation and the rest of strategies.

In Figures S56-S57, mean BEDROC(20) values are represented for each target and then grouped by their protein family. The BEDROC(20) model from Equation 8 and confidence intervals on its predictions can be found in Table S18 and Figure S58 from the Supporting Information. Based on this model, the best performing families appear to be, in descending order: IC (Ion Channels), GR (G-protein coupled receptors), TR (Transporters), OE (Other Enzymes) and PK (Protein Kinases). The same ordering of cross-validation schemes by performance is observed in the target-wise analysis.

4.3.3 Prospective test

Table 8 gathers the performance metrics for the prospective validation. These are described using the model in Equation 9, see Table S19 from the Supporting Information. Using AUC and BEDROC(20), the random cross-validation strategy is overly optimistic whereas database and intermediate are overly pessimistic when compared to the prospective estimates ($p < 0.05$

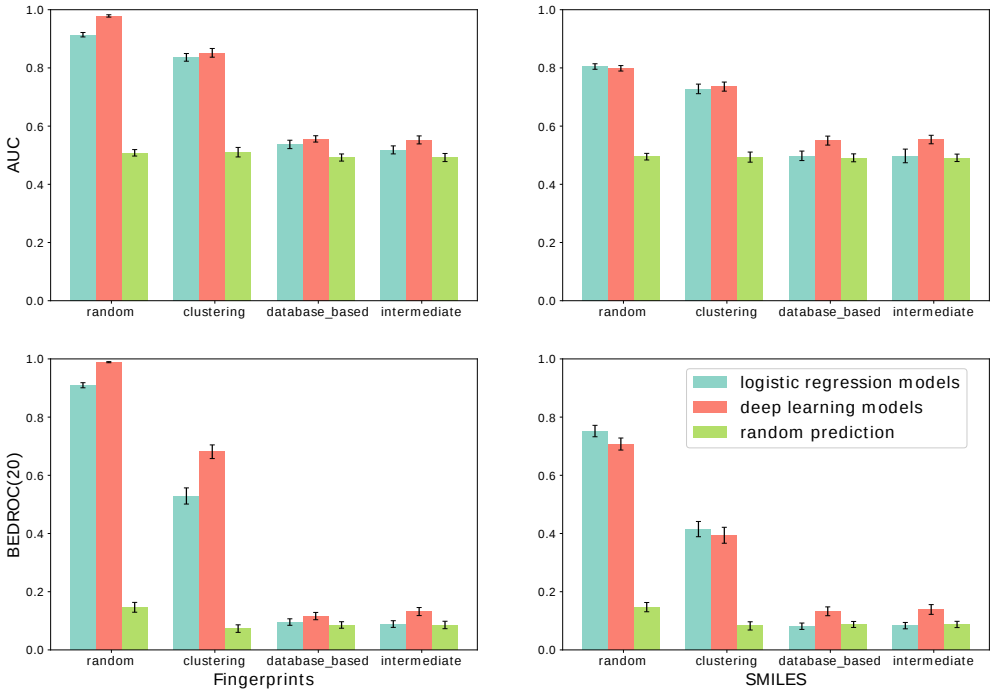

Figure 25: Comparison of performance of the different models, grouped by cross-validation strategies and colored by algorithm used for the prediction. Top row compares results in terms of AUC and bottom row in terms of BEDROC (α = 20) score. In blue, logistic regression models metrics are shown; in red, deep learning-based models, and in green, a set of random prediction synthetically generated. Left column shows results for fingerprints-encoded models and right column for SMILES-encoded models. Error bars indicate standard error of the mean from subsampled estimates.

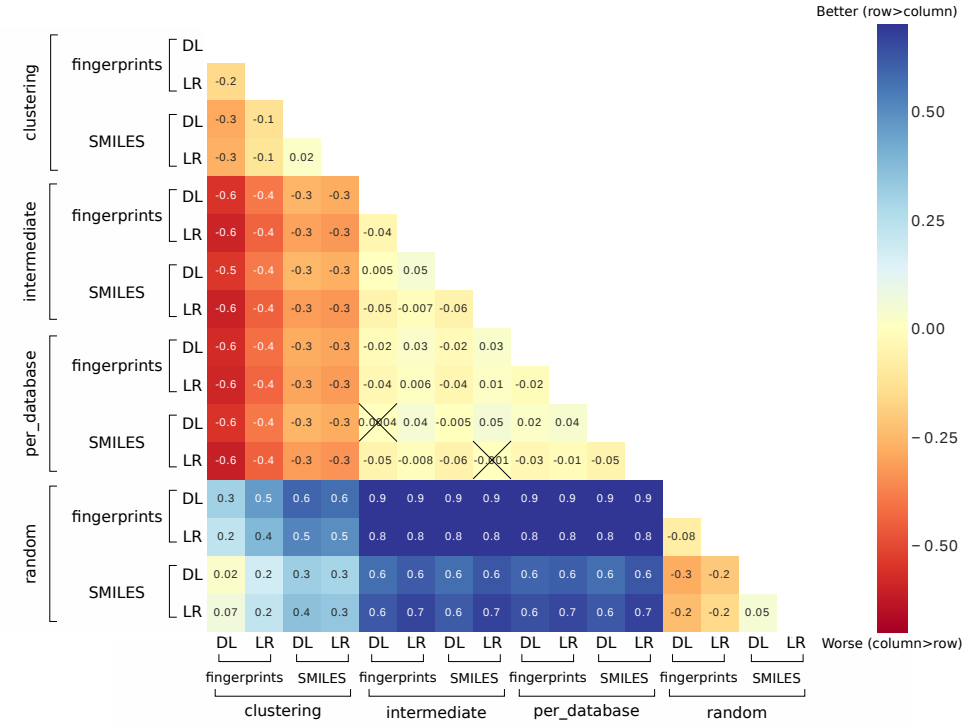

Figure 26: Mean difference for BEDROC(20) metrics between pairs of models. Differences are calculated subtracting column performances from rows. Crossed tiles indicate that difference for that pair of models is not statistically significant between the subsampling estimates (FDR < 0.05).

CV strategy	Encoding	Algorithm	F1 score	AUC	pAUC(5%)	BEDROC(20)	BEDROC(100)	κ
Random	Fingerprints	LR	0.509 ± 0.004	0.724 ± 0.003	$0.003 \pm 2.4 \cdot 10^{-4}$	0.634 ± 0.010	0.664 ± 0.035	0.303 ± 0.005
		DL	0.513 ± 0.005	0.634 ± 0.004	$0.005 \pm 2.8 \cdot 10^{-4}$	0.677 ± 0.010	0.699 ± 0.035	0.332 ± 0.006
	SMILES	LR	0.498 ± 0.004	0.707 ± 0.003	$0.004 \pm 1.9 \cdot 10^{-4}$	0.595 ± 0.009	0.596 ± 0.036	0.288 ± 0.006
		DL	$\mathbf{0.578 \pm 0.004}$	$\mathbf{0.797 \pm 0.003}$	$\mathbf{0.008 \pm 2.2 \cdot 10^{-4}}$	$\mathbf{0.734 \pm 0.007}$	$\mathbf{0.875 \pm 0.013}$	$\mathbf{0.385 \pm 0.005}$
Clustering	Fingerprints	LR	$\mathbf{0.552 \pm 0.005}$	0.753 ± 0.003	$4.2 \cdot 10^{-4} \pm 2.7 \cdot 10^{-4}$	0.641 ± 0.121	0.657 ± 0.041	0.355 ± 0.006
		DL	0.511 ± 0.004	0.640 ± 0.004	$0.005 \pm 3.0 \cdot 10^{-4}$	0.695 ± 0.010	0.725 ± 0.034	0.324 ± 0.005
	SMILES	LR	0.500 ± 0.005	0.712 ± 0.003	$0.003 \pm 1.6 \cdot 10^{-4}$	0.596 ± 0.009	0.624 ± 0.033	0.288 ± 0.006
		DL	0.547 ± 0.005	$\mathbf{0.782 \pm 0.003}$	$\mathbf{0.008 \pm 2.2 \cdot 10^{-4}}$	$\mathbf{0.733 \pm 0.007}$	$\mathbf{0.874 \pm 0.012}$	$\mathbf{0.356 \pm 0.006}$
Database based	Fingerprints	LR	0.440 ± 0.005	0.664 ± 0.004	$0.006 \pm 1.9 \cdot 10^{-4}$	0.650 ± 0.008	0.802 ± 0.014	0.258 ± 0.006
		DL	0.331 ± 0.004	0.577 ± 0.004	$0.006 \pm 1.6 \cdot 10^{-4}$	0.625 ± 0.007	$\mathbf{0.856 \pm 0.012}$	0.148 ± 0.005
	SMILES	LR	0.454 ± 0.004	0.627 ± 0.004	$0.005 \pm 2.1 \cdot 10^{-4}$	0.625 ± 0.009	0.677 ± 0.025	0.259 ± 0.005
		DL	$\mathbf{0.486 \pm 0.004}$	$\mathbf{0.768 \pm 0.003}$	$\mathbf{0.008 \pm 1.9 \cdot 10^{-4}}$	$\mathbf{0.716 \pm 0.007}$	0.850 ± 0.014	$\mathbf{0.280 \pm 0.005}$
Intermediate	Fingerprints	LR	0.436 ± 0.005	0.650 ± 0.004	$0.006 \pm 2.0 \cdot 10^{-4}$	0.662 ± 0.008	$\mathbf{0.810 \pm 0.015}$	0.249 ± 0.006
		DL	0.450 ± 0.004	0.604 ± 0.003	0.00 ± 0.00	0.497 ± 0.010	0.515 ± 0.038	0.153 ± 0.005
	SMILES	LR	0.442 ± 0.005	0.621 ± 0.004	$0.004 \pm 1.5 \cdot 10^{-4}$	0.605 ± 0.008	0.671 ± 0.027	0.238 ± 0.006
		DL	$\mathbf{0.477 \pm 0.005}$	$\mathbf{0.767 \pm 0.003}$	$\mathbf{0.008 \pm 2.1 \cdot 10^{-4}}$	$\mathbf{0.701 \pm 0.008}$	0.775 ± 0.015	$\mathbf{0.295 \pm 0.006}$

Table 8: Summary of performance on the prospective test set of the different cross-validation strategies for DL and LR models, in terms of F1 score, AUC, partial AUC (pAUC) at 5%, BEDROC(α=20) and BEDROC(α=100). Mean $\pm$ standard deviation of the subsampling estimates. In bold, the best performance for each CV strategy. CV: cross-validation.

in all cases adjusted by multiple testing, Tukey's method). The clustering strategy cannot be distinguished from the prospective validation in AUC ($p = 0.16$) and is slightly pessimistic for BEDROC(20) ($p = 0.0039$).

In Figure S59, the AUC and BEDROC(20) of the prospective test are compared between both algorithms trained under the four cross-validation strategies and with a random predictor. This Figure displays the behaviour seen on Table 8: differences between cross-validation strategies on the prospective test are not as pronounced as in Figure 25.

4.4 DISCUSSION

4.4.1 Validation type

We performed two types of validation in this study: the cross-validation design and the prospective one. The main thesis of this study was the estimation of the generalization of the models when controlling for the chemotype bias. For that purpose, cross-validation was modified to account for it and provide sensible estimates. On the other hand, provided that the prospective validation is not exempt of the chemotype bias (Kearnes et al., 2017; Wallach and Heifets, 2018), it was used to have an alternative performance estimate of each model in a common setting.

4.4.2 Cross-validation strategy

The choice of cross-validation strategy is the most influential factor in this study. In general terms, random cross-validation is over-optimistic. When adding constraints for set splitting, performance suffers a pronounced, significant drop. Such disagreements between cross-validation strategies suggest that compounds from different databases have different properties, i.e. that different molecular databases cover different regions of chemical space. This bias on data distribution would explain why models struggle to generalize between databases. On the other hand, any limitations derived from the

data selection and benchmark construction by Riniker and Landrum might affect the predictive power of the models.

Our results show that random cross-validation leads to the highest performance estimates for all the models. Likewise, random cross-validation shares the most proportion of compounds and proteins between the training, validation and test folds (Figures 22A and 22B). This is also true at the molecular scaffolds level (Figure 23) and regarding Tanimoto similarity between compounds from the test and the training set (Table 5), suggesting that shared scaffold and similarity between compounds inflate the performance estimates. This inflation can be clearly seen in the prospective validation, where the performance of the random strategy suffers a pronounced drop (BEDROC(20) = 0.677 ± 0.010 for the fingerprints DL model on the prospective validation versus BEDROC(20) = 0.956 ± 0.004 on the original test set). Despite this, random cross-validation has been traditionally used in literature to evaluate binding prediction models. Our results are in line with previous reports suggesting that the predictivity of most published virtual screening models is too optimistic (Kearnes et al., 2017; Wallach and Heifets, 2018).

Among the non-random cross-validation strategies, database-based and intermediate schemes lead to models barely outperforming a random predictor in terms of AUC and BEDROC(20), see Figure 25 and Table 7. These schemes are probably too conservative, because the protein families and their proportions in their training, validation and test folds differ (Figure 24). In addition, their number of scaffolds in train (respectively 10,344 and 8,947) is smaller than random (19,066) and clustering (17,648) strategies, see Figure 23, limiting the variety of examples the models can learn from the training data. This tendency is also seen in terms of Tanimoto similarity (Table 5): for intermediate and per_database, only 127 and 183 molecules in the test set have at least one similar molecule in the training set. Consequently, both schemes provide overly pessimistic estimates. For instance, the SMILES DL model has an estimated BEDROC(20) of 0.133 ± 0.007 and 0.139 ± 0.007 using database-based and intermediate strategies, whereas prospective validation reports 0.716 ± 0.007 and 0.701 ± 0.008 instead. Our results show that clustering on its own was not enough to overcome database bias, i.e. the model did not generalize between databases, which was the main reason for including the intermediate strategy.

The clustering strategy allows training with more scaffolds than database-based and intermediate strategies while keeping a similar proportion of protein families. It controls for the data redundancy issue as the number of scaffolds shared between train and test decreases from 2,676 (random) to 275. Equally, the number of molecules from the test set with one or more similar molecules in the training set is 3,273 for random cross-validation but only 236 for clustering. It also leads to less optimistic performance estimates than the random strategy, whereas the models still retain predictive power. The performance from this strategy cannot be distinguished from the prospective validation in AUC (see Tables S19 and S16), although being slightly more pessimistic for BEDROC(20) (see Tables S19 and S59). The latter might be an indicator of the presence of some residual chemotype bias in the prospective data. The clustering strategy proved to be a compromise

solution, in line with the prospective validation, and was therefore chosen as our reference, as in previous studies (Mayr et al., 2018).

4.4.3 Compound encoding

In general, models for compounds represented by their fingerprints outperformed models for SMILES representation in clustering-based and random strategies, for both AUC and BEDROC(20), see Figure 25 and Table 6. This can be due to the specific architectures employed in each case: the compound analysis block for SMILES-encoding is based on CNN and LSTM to capture the sequence structure, while the compound analysis block for fingerprints is based on a single feed-forward neural network. This difference in model complexity and by extension, in the number of parameters, could have resulted in a poorly fitted SMILES-encoded model. In the prospective validation this improvement is not as significant comparatively ($p > 0.1$ for both AUC and BEDROC(20) in the models from Equation 9, see Table S19 from the Supporting Information). However, given that fingerprints-based models show a better performance in the cross-validation experiments, we will focus on them from this point on.

4.4.4 Prediction algorithms

Regarding the algorithm used within the clustering strategy and FP encoding, most metrics suggest that DL outperforms LR: BEDROC(20) (0.684 ± 0.012 versus 0.529 ± 0.012), pAUC(5%) (0.024 ± 0.001 versus 0.015 ± 0.001), BEDROC(100) (0.779 ± 0.029 versus 0.532 ± 0.034), Cohen's Kappa (0.460 ± 0.011 versus 0.358 ± 0.011) and F1 Score (0.501 ± 0.010 versus 0.409 ± 0.011) (see Table 7). However, when describing all the cross-validation strategies and encodings (model in Equation 7), Table 6 shows that this improvement is negligible, if any ($0.05 \leqslant p < 0.1$ for AUC and $p > 0.1$ for BEDROC(20)), even though the logistic regression model is not optimal for the sequential inputs.

On the other hand, performance of the random cross-validation fingerprints-based DL model (BEDROC(20) of 0.956 ± 0.004) is similar to the other published DL-based proteochemometrics model (Lenselink et al., 2017) (0.962). Lenselink et al also apply a temporal split strategy that drops the BEDROC metric 0.114 units, while our clustering strategy penalizes 0.272 units to our DL model. This is expected as time-split cross-validation can still suffer from chemical series bias (Kearnes et al., 2017; Wallach and Heifets, 2018). Lenselink et al represent proteins through standard physicochemical descriptors, whereas we use their amino acid sequence. The fact that amino acid-based representations attain state-of-the-art power poses the opportunity to gain insights into the mechanisms causing protein-ligand binding by analyzing biological patterns in the CNN filters and the long-range dependencies in the LSTM.

4.4.5 Protein families

The imbalance in the distribution of targets between protein families (see Figure S41) make it difficult to extract conclusions about the differences in performance between families (Figures S56-S57, Table S18 of the Supporting information). The supposedly best performing family, IC, has only three targets, while the reference family (CY) and the third best performing (TR) have only one target each. After setting aside these families of small size, the best performing families are, in descending order: GR, OE and PK. These results are similar to the trend reported in the review by Qiu et al (Qiu et al., 2017): the increase of bioactivity data availability mostly happens to a small number of protein classes and disease targets, such as GR, PK or HIV-1 proteases (which would be classified as OE). This leads to less simulated data for these better-studied families and consequently, a positive effect in the prediction accuracy.

4.5 CONCLUSIONS

We have benchmarked protein-compound binding models using two molecular representations for compounds and two prediction algorithms under four cross-validation strategies. We have also compared theses strategies to a prospective validation.

One of our main findings is the existence of a database-specific bias that challenges the generalization of machine learning models between databases. Despite of being widely used in literature, performance estimates derived from classical random cross-validation are overly optimistic. Instead, we recommend a clustering-based cross-validation since it addresses the chemical series bias while providing more reliable performance estimates. For molecular representation, fingerprints have led to better models than the SMILES identification string. Regarding the prediction algorithm, deep learning models show residual improvement over a baseline logistic regression.

Although further interpretation of the CNN-LSTM architecture could provide valuable information on the binding mechanisms, our results highlight the importance of accompanying the Deep Learning model with a logistic regression baseline to quantify the added value of the deep architecture.

ACKNOWLEDGEMENTS

The authors thank the NVIDIA Corporation for the donation of the Titan Xp GPU used to perform the analysis of this article. This research was partially supported by an Industrial Doctorate grant from the Generalitat of Catalonia to A.L.-d.R. (DI 2016-080). This work was also supported in part within the framework of the Ministerio de Economía, Industria y Competitividad (MINECO) Grants TEC2014–60337–R and TEC2017 DPI2017-89827-R, and the Centro de Investigación Biomédica en Red (CIBER) of Bioengineering, Biomaterials and Nanomedicine, an initiative of the Instituto de

Salud Carlos III (ISCIII). The authors thank Sergio Picart-Armada for helpful discussions and statistical advice.

5 EFFECT OF PADDING SEQUENCES

EFFECT OF SEQUENCE PADDING ON THE PERFORMANCE OF DEEP LEARNING MODELS IN ARCHAEAL PROTEIN FUNCTIONAL PREDICTION

The use of raw amino acid sequences as input for deep learning models for protein functional prediction has gained popularity in recent years. This scheme obliges to manage proteins with different lengths, while deep learning models require same-shape input. To accomplish this, zeros are usually added to each sequence up to a established common length in a process called zero-padding. However, the effect of different padding strategies on model performance and data structure is yet unknown. We propose and implement four novel types of padding the amino acid sequences. Then, we analysed the impact of different ways of padding the amino acid sequences in a hierarchical Enzyme Commission number prediction problem. Results show that padding has an effect on model performance even when there are convolutional layers implied. Contrastingly to most of deep learning works which focus mainly on architectures, this study highlights the relevance of the deemed-of-low-importance process of padding and raises awareness of the need to refine it for better performance. The code of this analysis is publicly available at https://github.com/b2slab/padding_benchmark.

5.1 INTRODUCTION

Since the breakthrough of deep learning (DL) (Krizhevsky et al., 2012), deep neural networks are being successfully applied in computational biology (Angermueller et al., 2016; Eraslan et al., 2019). This is due to their capacity for automatically extracting meaningful features from raw data (LeCun et al., 2015). Specifically, DL is useful in the context of biological sequences, such as proteins or RNA, because it can learn directly from the sequence and hence, capture nonlinear dependencies and interaction effects. Some examples of applications of DL on biological sequences include prediction of specifities of DNA and RNA binding proteins (Alipanahi et al., 2015), DNA function quantification (Quang and Xie, 2016), de novo peptide design (Müller et al., 2018), detection of conserved DNA fragments (Yi Li et al., 2017), prediction of protein associated GO terms (Rifaioglu et al., 2019) or

This chapter is a postprint of the following journal article: Angela Lopez-del Rio, Maria Martin, Alexandre Perera-Lluna and Rabie Saidi. "Effect of Sequence Padding on the Performance of Deep Learning Models in Archaeal Protein Functional Prediction". *Scientific Reports, 10, 14634 (2020)*

quantification of the impact of genetic variation on gene regulatory mechanisms (Eraslan et al., 2019). The specific DL architectures able to leverage the inner structure of sequential biological data are Convolutional Neural Networks (CNN) and Recurrent Neural Networks (RNN). CNNs entail translational invariance (Kauderer-Abrams, 2017) and can be used to find relevant patterns with biological meaning (Alipanahi et al., 2015; Evans et al., 2018; Yi Li et al., 2017; Öztürk et al., 2018a). For their part, bidirectional RNNs (and the derived Long Short-Term Memory and Gated Recurrent Units) are appropiate for modelling biological sequences since they are suited for data with a sequential but non-causal structure, variable length, and long-range dependencies (Almagro Armenteros et al., 2017; Di Lena et al., 2012; Hochreiter et al., 2007; Lopez-Del Rio et al., 2019). Both architectures are usually combined, as in DEEPre (Yu Li et al., 2018), where a CNN-RNN model performs a hierarchical classification of enzymes.

Proteins are long linear sequences constituted by amino acid residues attached covalently. These amino acid residues are represented by letters that cannot be directly processed by the mathematical operations used by DL models. Choosing how to digitally encode amino acids is a crucial step in this context, since it can affect to the overall performance of the models (Domingos, 2012). A comprehensive review and assessment on different amino acid encoding methods (Jing et al., 2019) shows that position specific scoring matrix (PSSM), an evolution-based position dependent methodology, achieves the best performance on protein secondary structure prediction and protein fold recognition tasks. However, this type of encoding is very consuming computationally (Ahmad and Sarai, 2005) and its applicability is limited to proteins with known homologous sequences (Jing et al., 2019), which could highly decrease the generalisation capabilities of the predictor for non evolutionary related proteins. Traditionally, proteins have also been encoded into feature vectors (Lenselink et al., 2017; Strömbergsson et al., 2010). These encoding features are generally aggregative and not bijective, such as signatures, physicochemical properties or amino acid composition. From aggregative features, the original sequence cannot be recovered, resulting in a loss of protein information.

The analogy between text and proteins, understood as sequences of characters with a meaning, has motivated the application of Natural Language Processing (NLP) techniques to amino acid sequences. Along these lines, machine-learning derived embeddings (Asgari et al., 2015; Kimothi et al., 2016; Mazzaferro, 2017; K. K. Yang et al., 2018) and one-hot encoding (Jurtz et al., 2017; Yu Li et al., 2018; Lopez-Del Rio et al., 2019; Müller et al., 2018; Öztürk et al., 2018a; Rifaioglu et al., 2019) have become very popular. Specifically, the latter method has been widely used in protein-based DL models since neural networks are able to extract features from raw data. A schematic explanation of one-hot encoding is shown in Fig. 27B. Every amino acid of a protein sequence is represented by a binary vector of length $n+1$, n being the number of different amino acids and placeholders. In this vector, all but the corresponding entry for that amino acid is set to zero. As a result, a protein of length L is represented by a $(n+1) \times L$ binary matrix.

The main problem of one-hot encoding is that each protein has a different length, while all input vectors should be of the same size to be fed into the

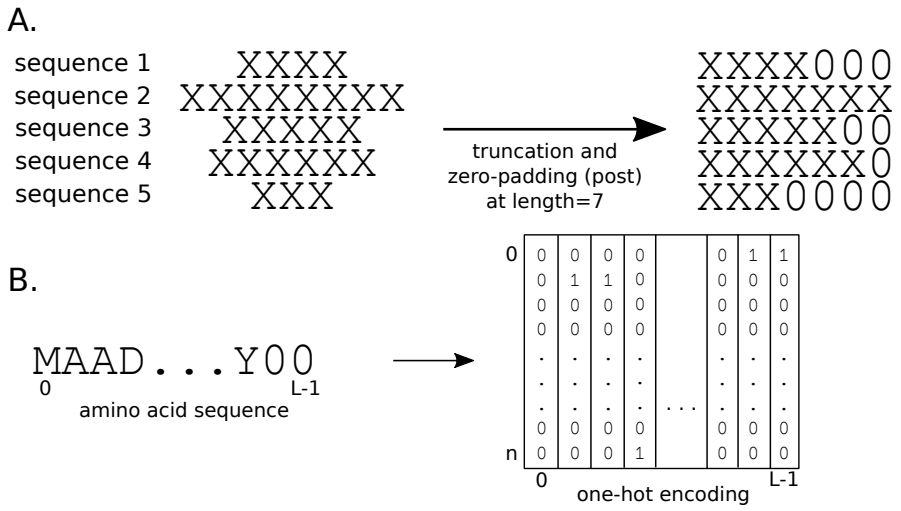

Figure 27: Schematic explanation of one-hot encoding, zero-padding and truncation of amino acid sequences A. Amino acid sequences of different lengths are shaped to the common dimension of 7 by truncating or padding zeros at the end. **B.** Amino acid sequence at common length L is transformed to a binary matrix $(n+1) \times L$, being n the number of different amino acids and placeholders. Each column of this matrix is full of zeros, being one only in the position of the corresponding amino acid.

model. To overcome this issue, sequence padding and truncation are usually applied (Lopez-Del Rio et al., 2019; Müller et al., 2018; Öztürk et al., 2018a; Rifaioglu et al., 2019). This means establishing a common length for all proteins and then, truncating longer proteins to that length or filling shorter proteins with an "artificial" character up until that length (see Fig. 27A). This process of completing a sequence is called padding and the character used for filling could be any that is not used in the sequences themselves. To this matter, zero character ("0") is the most commonly used. Padding zeros can be added at any position of the sequence, for example at the N- and C- terminals of the sequences (Mirabello and Wallner, 2019). In practice, they are usually added at the end (Lopez-Del Rio et al., 2019; Müller et al., 2018). However, details on the concrete steps of sequences padding are often omitted as they are deemed of low importance for the results of the study (Jurtz et al., 2017; Yu Li et al., 2018; Öztürk et al., 2018a; Rifaioglu et al., 2019). Even when this information is given, there is no proper justification on the padding choice (Lopez-Del Rio et al., 2019; Mirabello and Wallner, 2019; Müller et al., 2018). This is partly due to the lack of exhaustive studies on the effect of padding the sequences. Up to our knowledge, the work of Reddy et al, 2019 (Dwarampudi and Reddy, 2019) is the only study on the effect of sequence padding on deep learning models. It was applied on a NLP sentiment analysis task and only pre- and post- padding types were tested. Since it is a different application domain and the options they test are limited, a more comprehensive study for the case of biological sequences is needed. Likewise, alternative types of padding to those usually implemented (zeros at the end of the sequence, at the beginning or both) have not been yet explored.

Domains of application involving recurrent neural networks also make use of mask layers, in order to inform the model to skip the padding positions in the objective function and gradients (Parikh et al., 2016). However, masking lacks general support for convolutional, feed-forward, flatten or pooling layers. Since many of the amino acid sequence models in the literature contain some of these layers (Jurtz et al., 2017; Yi Li et al., 2017; Lopez-Del Rio et al., 2019; Öztürk et al., 2018a; Rifaioglu et al., 2019), and considering that recurrent layers have been proven not to always be the best choice in sequence-based models (Bai et al., 2018), it is still important to evaluate the potential effects of padding.

In this paper, we report a systematic analysis on how different types of padding affect to protein-based DL models performance. We evaluate this effect on three different DL architectures: only feed-forward neural networks (only_denses), feed-forward neural networks coupled with a convolutional layer (1_conv) and feed-forward neural networks coupled with a stack of convolutional layers (stack_conv). We also introduce four novel padding types (mid-, strf-, rnd-, and zoom-) and we classify them along with the known types (pre-, post- and ext-) into dense and sparse paddings. Dense paddings are those keeping zeros together in a block (pre- at the beginning, post- at the end, mid- in the middle and ext- at both ends), while in sparse paddings, zeros are interspersed on the sequence (randomly in the case of rnd- and uniformly for strf-) or amino acids are duplicated (zoom-). Finally, we quantify the effect for each type of padding. The chosen task for this study is a hierarchical classification of enzymes with two levels: the first is a binary classification of proteins into enzymes/non-enzymes (task 1), and the second is a multi-label prediction of the enzyme type (task 2).

5.2 RESULTS

5.2.1 Performance metrics

A summary of the F1-score (macro average), accuracy and Area under the receiver operating characteristic curve (AUC from now) on test for each architecture, each type of padding and each task (only task 1 in the case of AUC) is shown in Table 9. Since the trends observed for these metrics are analogous, we will focus on F1-score. Fig. 28 shows the macro F1-score on test for each type of padding in each of the tested architectures, both for task 1 and task 2. The same figure but for accuracy can be found in Fig. S60 of the Supporting Information. Figures S61-S66 show F1-score results per label (non-enzyme/enzyme in the case of task 1 and 1-7 enzyme types in the case of task 2) for each task and each of the architectures.

According to Table 9 and Fig. 28, although the different architectures seem to have similar F1-score values, only_denses architecture is the one that achieves the best performance for task 1 (paired Wilcoxon test, two-sided, p= 9e−15 vs 1_conv and p= 4e−13 vs stack_conv). Regarding task 2, we can see at Fig. 28 that the trend is not as clear as in task 1. 1_conv has the best performance (paired Wilcoxon test, two-sided, p= 9e−4 vs only_denses,

	Padding type	1_conv	only_denses	stack_conv
Task 1 F1-score	aug	0.756 ± 0.041	0.900 ± 0.011	0.790 ± 0.022
	ext	0.842 ± 0.022	0.896 ± 0.013	0.875 ± 0.010
	mid	0.868 ± 0.016	0.911 ± 0.018	0.874 ± 0.026
	post	0.873 ± 0.010	0.900 ± 0.014	0.879 ± 0.024
	pre	0.858 ± 0.011	0.899 ± 0.013	0.863 ± 0.028
	rnd	0.786 ± 0.014	0.896 ± 0.025	0.812 ± 0.011
	strf	0.867 ± 0.008	0.930 ± 0.011	0.851 ± 0.010
	zoom	0.868 ± 0.005	0.893 ± 0.021	0.862 ± 0.014
Task 2 F1-score	aug	0.536 ± 0.025	0.531 ± 0.022	0.504 ± 0.026
	ext	0.554 ± 0.022	0.543 ± 0.034	0.540 ± 0.023
	mid	0.558 ± 0.021	0.542 ± 0.027	0.557 ± 0.024
	post	0.550 ± 0.025	0.509 ± 0.075	0.554 ± 0.024
	pre	0.541 ± 0.020	0.541 ± 0.030	0.527 ± 0.028
	rnd	0.452 ± 0.026	0.448 ± 0.034	0.455 ± 0.020
	strf	0.550 ± 0.024	0.548 ± 0.026	0.547 ± 0.024
	zoom	0.543 ± 0.019	0.456 ± 0.063	0.515 ± 0.026
Task 1 Accuracy	aug	0.758 ± 0.037	0.901 ± 0.011	0.790 ± 0.023
	ext	0.843 ± 0.023	0.896 ± 0.014	0.875 ± 0.010
	mid	0.868 ± 0.016	0.911 ± 0.018	0.874 ± 0.027
	post	0.873 ± 0.010	0.900 ± 0.014	0.879 ± 0.024
	pre	0.858 ± 0.011	0.899 ± 0.013	0.863 ± 0.028
	rnd	0.790 ± 0.013	0.897 ± 0.025	0.815 ± 0.011
	strf	0.868 ± 0.008	0.930 ± 0.011	0.852 ± 0.010
	zoom	0.869 ± 0.005	0.893 ± 0.021	0.863 ± 0.014
Task 2 Accuracy	aug	0.548 ± 0.009	0.536 ± 0.012	0.527 ± 0.015
	ext	0.539 ± 0.016	0.544 ± 0.025	0.549 ± 0.017
	mid	0.539 ± 0.017	0.545 ± 0.018	0.569 ± 0.011
	post	0.530 ± 0.011	0.532 ± 0.024	0.560 ± 0.016
	pre	0.532 ± 0.009	0.542 ± 0.016	0.545 ± 0.018
	rnd	0.455 ± 0.011	0.473 ± 0.020	0.509 ± 0.017
	strf	0.556 ± 0.021	0.556 ± 0.012	0.565 ± 0.015
	zoom	0.550 ± 0.014	0.528 ± 0.021	0.531 ± 0.010
Task 1 AUC	aug	0.859 ± 0.021	0.951 ± 0.010	0.891 ± 0.013
	ext	0.927 ± 0.011	0.966 ± 0.003	0.949 ± 0.005
	mid	0.945 ± 0.009	0.972 ± 0.006	0.952 ± 0.010
	post	0.945 ± 0.007	0.969 ± 0.003	0.956 ± 0.007
	pre	0.935 ± 0.006	0.967 ± 0.004	0.949 ± 0.008
	rnd	0.871 ± 0.011	0.946 ± 0.014	0.891 ± 0.009
	strf	0.939 ± 0.006	0.978 ± 0.003	0.927 ± 0.006
	zoom	0.937 ± 0.004	0.978 ± 0.005	0.944 ± 0.005

Table 9: Summary of F1-score (macro average), accuracy and AUC on the test set. Results are reported for all the different types of padding for both task 1 and task 2 in each one of the tested architectures (except AUC, which is only available for Task 1). Mean ± standard deviation of the 10 folds.

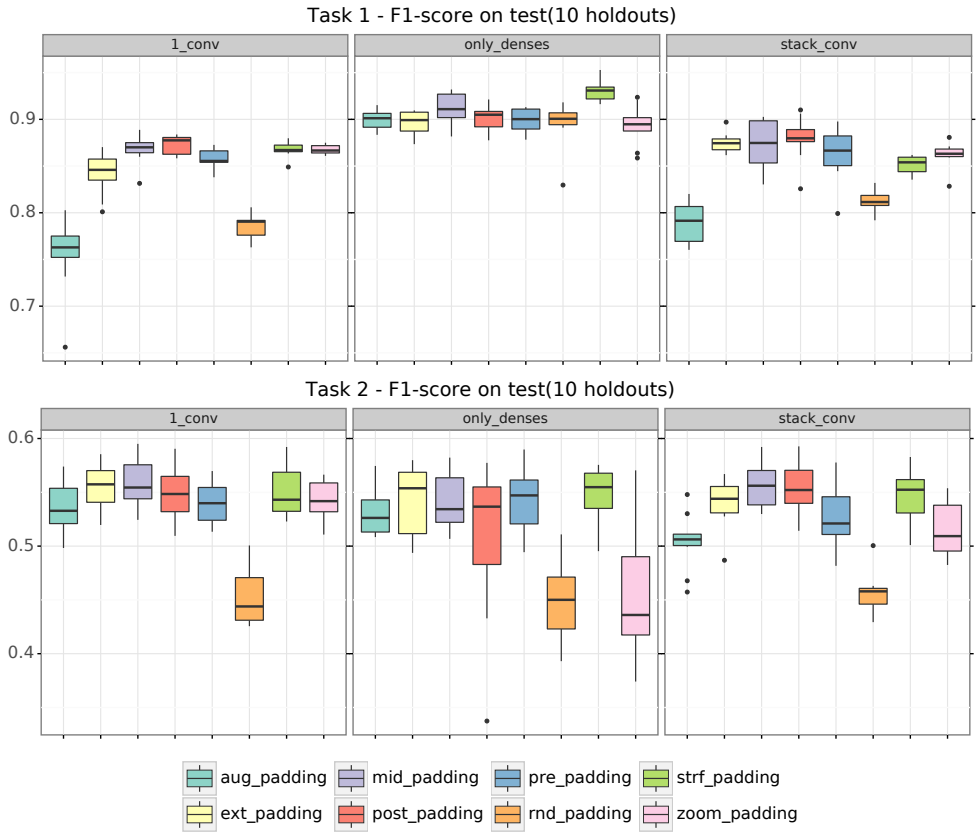

Figure 28: Macro F1-score on test for each type of padding in each tested architecture. Each boxplot comprises 10 data points (which is the number of folds).

p= 2e−5 vs stack_conv), while there are no statistical differences between stack_conv and only_denses.

Regarding metrics per label, in task 1 (Figures S61, S63 and S65) best recall results for non-enzymes were achieved in convolutional architectures, but the opposite trend is shown for the baseline architecture (only_denses). For task 2, classification of enzyme types 1, 4 and 6 achieved lower performance than 2 and 5. This applies to the three architectures (Figures S62, S64 and S66). As for enzyme type 7, results show high variability in comparison with the other types due to the limited number of samples of this class.

Effect on input space

We studied the distribution of the activations of the 1D Convolutional layer for the 1_conv model to analyse the effect of the padding type in the input space by means of a Principal Components Analysis (PCA). Figure 29 displays the density plot showing the principal components (PC) 1 and 2 of the activations from the 1D convolutional layer of the 1_conv architecture for each type of padding on each fold in task 2. In Fig. S67 of the Supporting Information, the same representation for task 1 is shown. Focusing on Fig. 29 for task 2, the distribution of the activations is very similar for dense

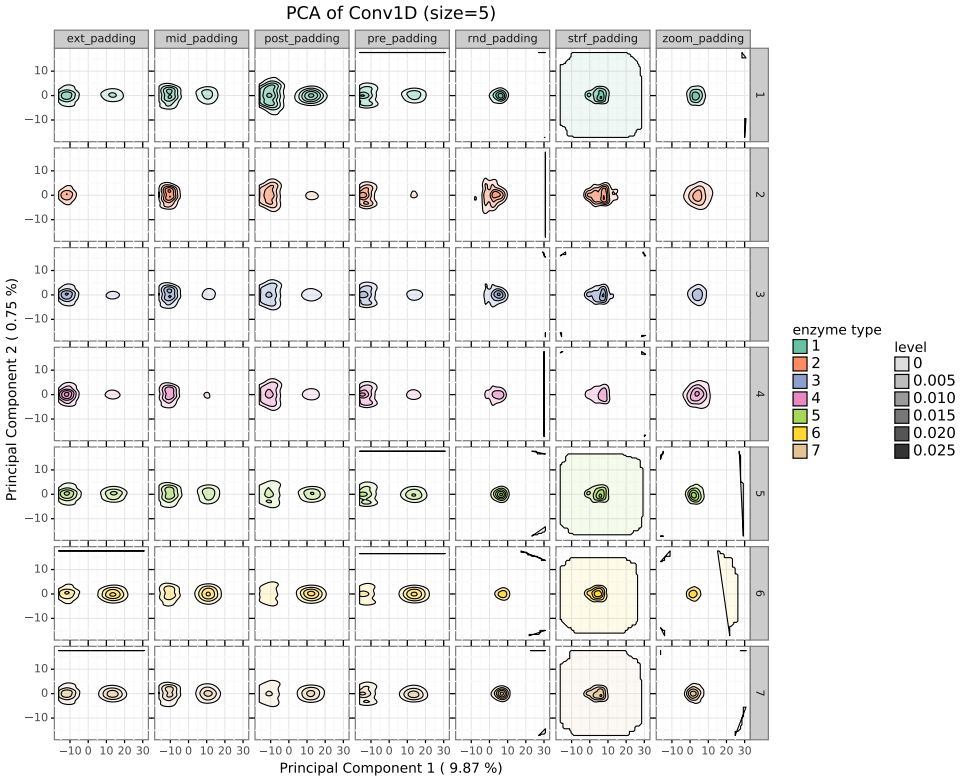

Figure 29: Density representation of the PCA of the activations of the convolutional layer. Figure shows PC1 vs PC2 of the activations of the 1D Convolutional layer for the 1_conv model in task 2, for the 10 folds. This representation comes from applying a PCA to the convolutional filter activations after the prediction of 14 enzymatic sequences of different EC number using each padding type. Then, the graphical representation was stratified by padding and enzyme type. We can see that according to the structure of the distributions, there seems to be two different groups of enzymes: 2, 3, 4 and 1, 5, 6, 7. Regarding types of padding, the activations for dense paddings are similar between them (two clusters separated along PC1) and different from sparse paddings.

types of padding (ext-, mid-, post- and pre-). These dense activations are grouped in two clusters separated along the PC1. Sparse paddings (rnd-, strf-, zoom-) activations have a distribution very different to that from dense paddings. In this case, activation points are condensed in one area, although each one of these types of padding has its own structure. Regarding enzyme types, according to the structure of the distributions, there seems to be two different groups: enzyme types 2, 3 and 4 are very similar between them and in turn, different to types 1, 5, 6 and 7. Table S28 of the Supporting Information quantifies the effect of the enzyme and the padding types on the PC1 of the activations using a linear model. All the terms of the model are significant.

5.2.2 Explanatory models

We used linear models to further explain the performance metrics and the effect of different variables (padding type, enzyme type, architecture) to the DL models behaviour. These explanatory models were also used to address specific questions regarding the effect of padding.

The full additive linear model in Equation 10 describes the F1-score values on test and it is shown on the Table S22 of the Supporting Information. It shows that some types of padding have indeed an effect on models performance, both for task 1 and 2. For example, for all the architectures (since it is an additive model) in task 1, aug-, pre-, ext- and rnd- have worse performance when compared to the reference padding type (post-) (p-value $<$ 0.05). In the same setting but for task 2, rnd- and zoom have significantly worse performance than post- (p-value $<$ 0.01).

Figure 30 and Tables S23-S27 gather the answers to our specific questions on the effect of padding on the different architectures and enzyme types. The colour represents in each case the difference between each category and the reference category of that factor. The sign of the corresponding estimate is represented if that difference is statistically significant. The constant term of a model (Intercept) shows the prediction when all the categorical variables have their reference values.

A. Does padding position affect performance?

Figure 30 A and Table S23 show that the classification performance of the baseline model (only_denses) for task 1 is the same for all the types of dense padding, except for strf-padding, which is better than post- (predicted F1-score of 0.926 vs 0.896). In the same way for task 2, only strf- significantly outperforms post-padding (0.472 vs 0.432), while zoom- has a worse performance (0.379).

B. What is the effect of switching between dense paddings?

Figure 30 B and Table S26 for task 1 show that for stack_conv, there are no differences in performance between dense paddings. Regarding task 2, only pre-padding is significantly worse than post-padding (0.457 vs 0.484) for stack_conv.

C. What is the effect of changing from the standard dense padding to sparse padding?

Figure 30 C and Table S25 show that for stack_conv in both tasks, sparse paddings have significantly worse performances than dense paddings. For task 1, post- significantly outperforms rnd-, strf- and zoom-padding (predicted F1-scores of 0.868 vs 0.801, 0.840 and 0.851, respectively). For task 2, post- also outperforms rnd- and zoom-padding (0.498 vs 0.400 and 0.460, respectively).

D. Is an ensemble of paddings beneficial?

We tested for the three architectures if data augmentation regarding padding (aug-padding) improved the performance when compared to post-padding (representing the dense paddings) and strf-padding (representing the sparse paddings). To quantify the changes, we used aug-padding as reference level in padding type. Figure 30 D and Table S26 show that both post- and strf- significantly outperform aug-padding both for task 1 and task 2.

For task 1, with the baseline architecture (stack_conv) aug-padding gets the worst predicted F1-score (0.786), while it is 0.875 and 0.847 for post- and strf- respectively. For the reference padding type (aug-), the stack_conv architecture performs worse than only_denses (0.785 vs 0.895) but better than 1_conv (0.785 vs 0.752). Interactions show that both strf- and post- have a more positive effect on performance with respect to the baseline (aug-) for 1_conv (0.863 and 0.868 vs 0.752 from aug-) than for stack_conv (0.847 and 0.875 vs 0.786). On the contrary, changing from aug- to strf- and post- have less performance improvement for only_denses (0.895 from aug- vs 0.926 and 0.896) than for stack_conv (0.786 vs 0.847 and 0.875), although even so only_denses still outperforms stack_conv.

Regarding task 2, with the baseline architecture aug-padding also gets the worst performance when compared to post- and strf-padding (0.421 vs 0.471 and 0.464). For baseline padding type, both 1_conv and only_denses significantly outperform stack_conv (0.453 and 0.449 vs 0.421). Interactions show that 1_conv reacts the same way to changes of padding type than stack_conv. But in the case of only_denses, changing from aug- to post-padding (0.449 vs 0.426) has less performance improvement than for stack_conv (0.421 vs 0.471), causing aug- to outperform post-.

E. Does the effect of padding type on performance depends on the enzyme type?

We checked for all the types of architectures and for dense and sparse paddings (represented by post- and strf-, respectively) the effect of enzyme type on model performance (only for task 2). Results (Figure 30E and Table S27) show that for both padding types, the performance for enzymes with the first EC number digit 2 (0.619), 3 (0.555), 5 (0.598), 6 (0.532) and 7 (0.597) is better than for digits 1 (0.457) and 4 (0.454). Interactions are not significant, meaning this trend applies to all the architectures. This is consistent with the results of the previous questions, where enzyme types 4, 1 and sometimes 6 are shown to decrease performance.

5.3 DISCUSSION

It is not the aim of this paper to study differences on performance between architectures. However, in general terms only_denses has shown to achieve the best performance for task 1 while both convolutional architectures work better for task 2 (see Tables 9, S22-S25 and S27, and Fig. 28-30). Quantitatively, for the full additive model (eq. 10) in task 1, only_denses

get a predicted F1-score of 0.916 vs 0.864 of stack_conv and 0.853 of 1_conv for the baseline post-padding type. Using the same model for task 2, predicted F1-score of only_denses is 0.444 versus 0.454 of stack_conv and 0.464 of 1_conv. The reason why only_denses is the best architecture for task 1 could be that the task of classifying amino acid sequences between enzymes and non-enzymes is more related to the presence/absence or count of certain amino acids than to their position within the sequence. In other words, if we could consider them to be amino acid sets instead of sequences as it happens in other fields (Vinyals et al., 2016). On the contrary, classifying enzymes into their types is a more complex task that might imply amino acid patterns and position information, thus a convolutional architecture is a better choice.

Along the same lines, we have seen that task 1 has a better performance than task 2 for all the architectures (Table 9 and Fig. 28). F1-score ranges from 0.756 ± 0.041 to 0.930 ± 0.011 for task 1, while for task 2 it is comprised between 0.448 ± 0.034 and 0.558 ± 0.021. Task 1 results are similar to those obtained by DEEPre (Yu Li et al., 2018) for their equivalent Level 0 prediction, but results obtained for task 2 are worse than their report for Level 1. This was expected, since we use the same architecture for a simple binary classification and for a multi-class classification problem. A more complex, optimized model may improve the performance for the first digit prediction problem, but this was out of the scope of this study. We chose the architectures of both tasks to be as simple, comparable and interpretable as possible.

We have confirmed that padding type has an effect on model performance (see Tables S22-S27 and 9, Fig. 30). In Fig. 29 and S67 we could see that indeed, models reflect differences for each type of padding in their input space.

In general, there are no differences between dense paddings (see Fig. 30 A and B and Tables S23-S24), neither for convolutional nor for only_denses architectures. This applies for both task 1 and task 2, although for the latter pre-padding underperforms the rest of dense paddings (predicted F1-score 0.457 from pre- vs 0.484 from post-). Therefore, dense paddings are interchangeable for fully feed-forward and dense architectures and we could stick to the default option (post-padding).

There are differences between sparse paddings. For the baseline model (only_denses) in both tasks (Figure 30 A and Table S23), strf_padding has shown to outperform the rest of the paddings: for task 1, strf- has a predicted F1-score of 0.926 vs 0.896 from post-; in task 2, strf- has an estimate of 0.472 while for post- it is 0.432; macro-average for the F1-score on test is 0.930 ± 0.011 for task 1 and 0.548 ± 0.034 for task 2 (Table 9) . This might be because feed-forward neural networks are position-sensitive and moving a block of zeros along the sequence (as in different types of dense padding) can alter the way the networks process them. Strf- does not comprise a block of zeros, but they are spread uniformly along the sequence. This distribution seems to compensate this position sensitivity by aligning certain relative positions of the protein where the model might be detecting abundance changes.

On the contrary, this improvement of performance caused by strf-padding does not apply for stack_conv architecture (Figure 30 C and Table S25). In this case, all sparse paddings perform worse than the baseline post-padding

(except for strf- in task 2): for task 1, the predicted F1-score of post-padding is 0.868 vs 0.840, 0.801 and 0.851 from strf-, rnd- and zoom- respectively; for task 2, predicted F1-score is 0.498 for post- vs 0.400 and 0.459 for rnd- and zoom-. Thus, convolutional models works better with dense paddings than with sparse ones.

The differences in activations of the convolutional layers in Fig. 29 further support the classification of paddings into dense and sparse and are in line with the results that we have just reported. The activations for the dense paddings showed to be very similar between them. This is expected due to the translational invariance of the convolutional layers (Kauderer-Abrams, 2017): if zeros are kept together they should be processed in the same way by the convolutional layers, no matter where they are located. In turn, the activations of dense paddings are very different from the sparse ones (Figure 29). Sparse paddings have also a similar structure, where the activations are condensed in only one centered group.

We have also tested if data augmentation regarding padding (i.e. artificially increasing the size of a dataset by representing one protein by different possibilities of the padded one-hot encoded amino acid sequence) improved model performance as in image deep learning models (Shorten and Taghi M. Khoshgoftaar, 2019). Our results (Figure 30 D and Table S26 of the Supporting Information) have shown that aug-padding underperforms dense and sparse paddings both for fully dense and convolutional architectures and for both tasks: for stack_conv task 1 aug- achieves a predicted F1-score of 0.786 vs 0.875 and 0.847 from post- and strf-, respectively; for task 2, 0.421 from aug- vs 0.470 from post- 0.464 and from strf-. In Fig. 28 it also shows to have the worst performance in both tasks for convolutional architectures. Hence, an ensemble of mixed dense and sparse paddings does not improve the performance of the models in this case. Augmented data using only sparse paddings or only dense paddings might work better, because then sequences would be in similar activation spaces.

We observed that models underperformed in enzyme types 1 (oxidoreductases) and 4 (lyases). This was noticeable by displaying the raw metrics (Figures S62, S64 and S66) and further confirmed through the explanatory models (Figure 30 and Table S27, the predicted F1-scores for enzymes 2, 3, 5, 6 and 7 are 0.619, 0.555, 0.598, 0.532 and 0.597, respectively, while it is 0.457 and 0.454 for 1 and 4). It does not seem to be related to the number of samples (Figure S68 of the Supporting Information), to sequence length (see Figure S69 of the Supporting Information) or to the distribution of the activations (Figure 29). Therefore, we assume that this is caused because these enzyme types are inherently more difficult to classify, as it happens in H. B. Shen and K. C. Chou, 2007. EC number prediction can be challenging in some cases due to divergent evolution (two enzymes with a completely different EC may actually be very similar in sequence) (Furnham et al., 2012) and parallel evolution of enzyme function (two completely unrelated enzymes catalyse the same reaction and thus, share EC number) (Holliday et al., 2012).

In Fig. 29 there also seems to be two groups of enzymes according to the distribution of the activations: 1, 5, 6 and 7 vs 2, 3 and 4. This could be partly related to the sequence length: Figure S69 of the Supporting In-

formation show that enzyme types 2, 3 and 4 are shorter than 1, 5, 6 and 7 ($p = 9e{-}54$ for Mann-Whitney-Wilcoxon test for independent samples, two-sided); moreover, these differences are not so visible for zoom_padding, for which models cannot count zeros. On the other hand, table S28 of the Supporting Information reports negative coefficients for enzyme types 2, 3 and 4, and positive coefficients for enzymes 5, 6 and 7 (enzyme type 1 is the reference) in the explanatory linear model for PC1, which further supports this grouping.

The results of this study have been obtained for amino acid sequences. It would be needed as a future work to investigate if this effect of padding on model performance can be translated to other biological sequences that are also one-hot encoded and padded, such as RNA (Pan and H.-B. Shen, 2018; Shrikumar et al., 2017) and miRNA (Zheng et al., 2019) or DNA sequences (Quang and Xie, 2016).

5.4 CONCLUSION

The effect of padding amino acid sequences when they are one-hot encoded had not been comprehensively addressed in the literature yet. The lack of this analysis has caused numerous studies to disregard this step, most of the times taking the "default" option and in some cases, even omitting the details around it. In this paper, we have shown that padding position has an effect on model performance.

We have tested seven types of padding using three different deep learning architectures in a hierarchical enzyme classification problem. It is the first study analysing the relevance of padding one-hot encoded amino acid sequences and its impact on the performance of the studied task.

Our results show that padding the amino acid sequence has an effect on the performance of models. Therefore, more attention should be given to this often omitted step of data pre-processing when building deep learning models for one-hot encoded proteins.

We propose and analyse novel ways of padding proteins when one-hot encoding them for machine learning models (strf-, zoom-, rnd-, mid-). Up until our knowledge, these types have been neither mentioned in the literature nor implemented and made publicly available. We provide the code for their application (`https://github.com/b2slab/padding_benchmark`), since we have shown that some of them could be more suited for their specific task or architecture.

Our results on EC number classification show that there are no differences between dense paddings. Thus, we can stick to the traditional post-padding, which has proved to outperform the other padding types for convolutional architectures. Regarding sparse paddings, our newly proposed strf-padding has shown to be the best choice for fully feed-forward neural networks, outperforming both dense paddings and the other types of sparse paddings. Lastly, data augmentation regarding the padding (aug-padding) does not improve performance. In contrast, it seems to add noise that causes performance to decrease.

	Non-enzymes	8,727
	Total enzymes	10,872
Enzyme type	1	1,187
	2	3,843
	3	2,123
	4	1,281
	5	603
	6	1,715
	7	120

Table 10: Distribution of UniprotKB/Swiss-Prot database proteins for taxonomy Archaea. Distribution is shown according to the enzyme type, which is determined by the first digit of the EC number. Entries without EC number are considered as non-enzymes.

This analysis has been applied to the specific task of EC number prediction. Although we cannot extrapolate these results to other tasks or other deep learning architectures, this is a starting point that highlights the need to avoid neglecting the padding step when one-hot encoding amino acid sequences, since we have shown that it has an effect on model performance.

5.5 MATERIAL AND METHODS

5.5.1 Material

Different types of padding were evaluated on the UniprotKB/Swiss-Prot database (UniProt, 2017) (version 2019_05) protein entries for taxonomy Archaea. For computational reasons we established an upper threshold of 1,000 amino acids for sequence length, leaving 19,385 proteins for training the models (more than 99% of the original entries). For the enzyme classification task performed for the padding analysis, Enzyme Commission number (EC number) annotation was used. EC number is a numeric classification schema for enzymes related to the chemical reactions they catalyze. Each EC number is constituted by 4 numbers separated by dots, being each one a progessively more specific classification. We only used the first digit of the EC number, which refers to the class of enzyme (1: oxidoreductases, 2: transferases, 3: hydrolases, 4: lyases, 5: isomerases, 6: ligases and 7: translocases) and considered the entries without EC number annotation as non-enzymes. 214 entries with more than one EC number were expanded as additional samples, having a total of 19,599 samples. Table 10 shows the enzyme type distribution of the dataset and Fig. S68 of Supporting Information represents this distribution. Data was divided 70/15/15% in training, validation and test sets. The training set was used to fit the model, the validation set was used to evaluate the model fit in each epoch and tune hyperparameters accordingly, and the test set was used to externally evaluate the final model fit. To check the consistency of the results, this splitting was randomly performed 10 times, so each model was trained and tested in each one of these data partitions.

5.5.2 Amino acids encoding and protein padding

Amino acids were represented by one-hot encoding. Seven different padding types were applied to those sequences shorter than 1000 amino acids (see Fig. 31): (I) **post-padding**, adding zeros at the end of the sequences; (II) **pre-padding**, adding zeros at the beginning of the sequence; (III) **mid-padding** (middle), adding the zeros in the middle of the sequence; (IV) **strf-padding** (stratified), distributing the zeros uniformly across the sequences; (V) **ext-padding** (extreme), adding zeros at both ends of the sequence in a balanced way (half of the padding pre- and half of the padding post-); (VI) **rnd-padding** (random), adding zeros at random positions of the sequence; (VII) **zoom-padding**, similar to stratified padding but instead of zeros, contiguous amino acids are repeated; this is the only padding type that "modifies" the sequence length. Additionally, (VIII) **aug-padding** (augmented) will assess the use of data augmentation regarding padding: each sequence will be represented by the seven different padding strategies.

We divided the types of padding in two groups: (1) **dense paddings**, those strategies that keep the sequence to a great extent, i.e. post-padding, pre-padding, ext-padding, mid-padding and (2) **sparse paddings**, which comprises those types of padding which repeatedly modify the structure of the sequence by inserting elements in between: strf-padding, rnd-padding and zoom-padding.

5.5.3 Classification task: hierarchical models

We tackled the enzyme classification task as a hierarchical problem with a level-by-level prediction strategy, as in Yu Li et al., 2018 (see Fig. 32), although we only approached the first two levels of the structure. This decision was taken due to the data imbalance (see Fig. S68 on the Supporting Information and Table 10) between non-enzymes and the less populated enzyme classes (e.g. class 7). We built two prediction models. Firstly, a binary classification model that, given a sequence, predicts if it is an enzyme or not. From now on, it will be referred as *task 1*. Secondly, a multilabel classification model with seven outputs that, given a sequence classified as enzyme by the first model, predicts the class of the enzyme (the first digit of the EC number). This will be referred as *task 2*.

5.5.4 Models architecture

We analysed the padding effect on three DL architectures: 1. a model with only feed-forward neural networks (it will be referred as *only_denses*), 2. a model with feed-forward neural networks and one 1D convolutional layer (*1_conv*) and 3. a model with feed-forward neural networks and five 1D convolutional layers stacked in parallel (*stack_conv*). The schematic representation of the four models can be found in Fig. S70 and S71. CNNs from the second and third model are aimed to detect meaningful patterns in the amino acid sequence. In all cases, dropout is used to prevent overfitting (N. Srivastava et al., 2014). The *only_denses* model was considered as

baseline or reference model, to have the simplest reasonable deep learning model to which we could compare against (Skiena, 2017) ; the *1_conv* model was chosen to study the effect of adding a convolution to the model and *stack_conv* was taken to check the effect on a convolutional architecture of relative complexity (Jurtz et al., 2017; Lopez-Del Rio et al., 2019). Further details of the models and the corresponding hyperparameters can be found in the Supporting Information file.

We tried to fit a bi-LSTM model to also test the effect of padding on this architecture. However, this model was too complex to converge within the range of parameters of the other three architectures (number of epochs, optimizer, learning rate). As stated by Li et al, 2018 (S. Li et al., 2018), LSTMs have convergence issues when training long sequences (length $\geqslant$ 1000). Because of this, we considered that the results of the bi-LSTM were not comparable to those from the other architectures and thus, decided to remove it from the analysis.

5.5.5 Implementation

Models were trained with an Adam optimizer (Diederik P Kingma and Ba, 2014) for 200 epochs, with a batch size of 54 (learning rate = 1E-4, $\beta_1 = 0.9$, $\beta_2 = 0.999$). Models were implemented in Python (Keras (Chollet et al., 2015) 2.2.4 using as backend TensorFlow (Martín Abadi et al., 2015) 1.8.0) and run on the GPU NVIDIA TITAN Xp and NVIDIA GeForce GTX 1070.

5.5.6 Performance metrics

The final model is that of weights corresponding to the epoch with the maximum validation accuracy in each case. Accuracy is the proportion of correct predictions. We tested each selected model on the corresponding test set of that data partition. For evaluating and comparing the performance of the different padding types, accuracy, F1-score for each label and macro F1-score on the test set were used. F1-score is the harmonic mean of precision (proportion of positive class predictions that actually belong to the positive class) and recall (proportion of correct positive class predictions out of all positive examples in the dataset). The macro F1-score is computed by calculating F1-score for each label and getting their unweighted mean, hence being insensitive to class imbalance. AUC was also computed for task 1 since it is a binary classification problem. AUC represents the probability that the classifier will rank a randomly chosen positive sample higher than a randomly chosen negative sample. Further details on the definition of these metrics can be found in the Supporting Information file. AUC was computed from raw predicted probabilities, while F1-score and accuracy were calculated from the binarized predictions at threshold 0.5. To statistically compare these metrics between architectures and types of padding, non-parametric two-sided Wilcoxon tests for paired samples were carried out (Wilcoxon, 1945).

5.5.7 Effect on input space

To analyse the effect of the padding type in the input space, we studied the distribution of the activations of the 1D Convolutional layer for the 1_conv model. This layer has sixty-four filters of size 5 (see Fig. S64 of the Supporting Information).

We randomly selected seven proteins of each type (for task 1) and two proteins of each type (for task 2) from the test set. For the two tasks separately, for each type of padding (except aug_padding) and each fold, we used the final model to predict on those proteins. We extracted the activations of the 1D convolutional layer for each prediction and separated each one of the sixty-four filters as a different sample. This resulted in a matrix of dimensions 64×1000 representing the activations for each prediction. Stacking the activation data of all the predictions (10 folds $\times 7$ types of padding $\times 14$ enzymatic sequences $= 980$ activations matrices of size 64×1000) separatedly for each task, we performed a PCA to study and compare the distribution of these activations.

5.5.8 Explanatory models

The performance metrics were further described through linear models built upon different variables that could affect to the model behaviour. These explanatory models have already been used for similar purposes (Lopez-Del Rio et al., 2019; Picart-Armada et al., 2019) and provide a way of statistically quantifying and comparing the relevance of the considered variables on the models performance.

Differences in performance between the different types of padding for both tasks were explained and tested in terms of the following linear model:

$$\text{F1} \sim \text{architecture} + \text{enzyme\_type} + \text{type\_padding} \tag{10}$$

This full additive model was used as a snapshot of the general contribution of each factor to the F1-score. Reference category for enzyme type was 0 (non-enzyme) for task 1 and enzyme type 1 for task 2, only_denses for architecture and post_padding for padding type. However, to answer more specific questions about the effect of padding on the different architectures and enzyme types, we built more precise, appropriate models in each case (Equations 12, 11 and 13). Some of them included an interaction term (represented by $var1 : var2$) to check if the effect of $var1$ on the F1-score depends on the value of $var2$. For example, adding $\texttt{type\_padding} : \texttt{architecture}$ would let us identify those cases where the effect of changing the type of padding is different between architectures. Reference category for enzyme type was still 0 for task 1 and enzyme type 1 for task 2, but for architecture and padding type it varies in each case. Table 11 summarizes the questions addressed through the explanatory models, their equations and the reference levels in each case. We considered only_denses as reference in Question A because it is the baseline model and we aimed to check if in this case, different paddings affect differently to model performance. In questions B-E, stack_conv is chosen as reference architecture since it is the more complex

Questions on the effect of padding addressed through the explanatory models.

	Question	Equation	Architecture	Padding type
A	Does padding position affect performance?	Eq. 11	**only_denses**	**post-**, pre-, mid-, ext-, strf-, rnd-, zoom-, aug-
B	What is the effect of switching between dense paddings?	Eq. 11	**stack_conv**	**post-**, pre-, mid-, ext-
C	What is the effect of changing from the standard dense padding to sparse padding?	Eq. 11	**stack_conv**	**post-**, strf-, rnd-, zoom-
D	Is an ensemble of paddings beneficial?	Eq. 12	**stack_conv** 1_conv only_denses	**aug-**, post-, strf-
E	Does the effect of padding type on performance depends on the enzyme type?	Eq. 13	**stack_conv** 1_conv only_denses	**post-**, strf-

Table 11: The explanatory model for each question is specified by the column Equation. The Architecture column and the Padding type column show the architectures and padding types included in each comparison, respectively. All the enzyme types are included for each question. Reference categories are indicated in bold.

and thus, the closest to the state of the art. Question E was only applied to task 2 results because its aim is to check the effect of enzyme types.

$$F1 \sim enzyme\_type + type\_padding \tag{11}$$

$$F1 \sim architecture + enzyme\_type + type\_padding + \\ + type\_padding : architecture \tag{12}$$

$$F1 \sim architecture + enzyme\_type + type\_padding + \\ type\_padding : enzyme\_type \tag{13}$$

Linear models were built in the R statistical programming language (R Core Team, 2015). P-values were adjusted for multiple testing by the False Discovery Rate (FDR) by Benjamini-Hochberg (Benjamini and Hochberg, 1995).

DATA AVAILABILITY

The UniprotKB/Swiss-Prot database (version 2019_05) protein entries analysed during the current study can be accessed and downloaded through the following link: http://ftp.ebi.ac.uk/pub/databases/uniprot/previous_releases/release-2019_05/knowledgebase/uniprot_sprot-only2019_05.tar.gz . Since this data needs further filtering to get only taxonomy Archaea, we have also uploaded data analysed in this article to the following repository: doi.org/10.6084/m9.figshare.11985750.

The code is publicly available at https://github.com/b2slab/padding_benchmark.

ACKNOWLEDGEMENTS

The authors thank the NVIDIA Corporation for the donation of the Titan Xp GPU used to perform some of the analysis of this article. The authors thank Sergio Picart-Armada for helpful discussions and statistical advice; Rossana Zaru and Antonio Ribeiro for helping with the biological/enzymatic interpretation of the results and Mahdi Mahmoudy for the technical help.

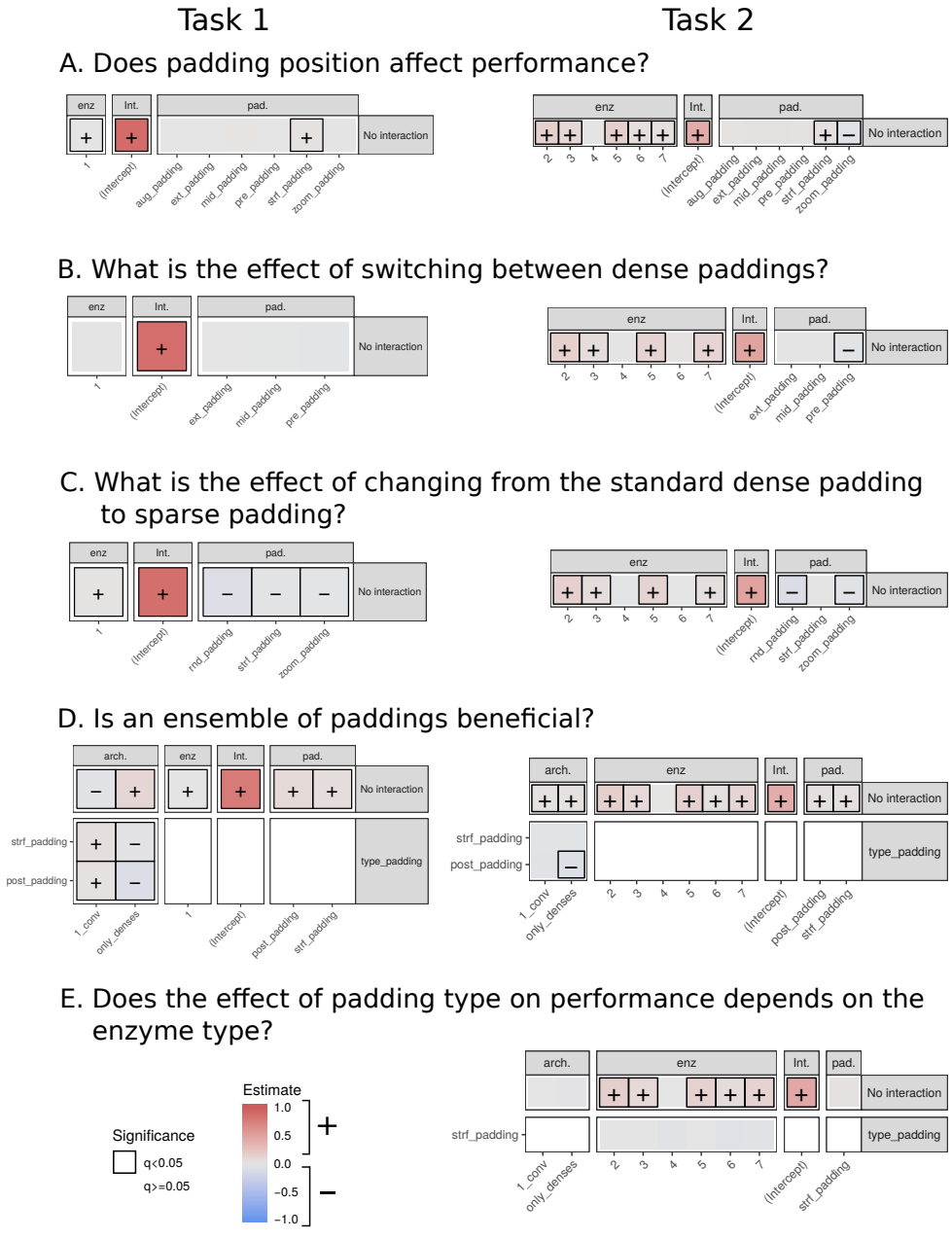

Figure 30: Linear models on F1-score for both tasks 1 and 2 focusing on five specific questions. The "No interaction" facet represent estimates of additive terms of the model, while the other facet represent the interaction between two factors. Models regarding questions A, B and C have no interaction terms and thus, they only have the "No interaction" facet. Only coloured tiles correspond to model coefficients; the white ones are outside the model specification. The colour of each category of a term represents the value of the estimate: red tiles correspond to positive estimates, specified with a "+", blue tiles correspond to negative estimates, specified by "-" and grey tiles are close to zero. Framed categories are those that have a significant effect on each question (adjusted p-value $<$ 0.05). An example on how to interpret this figure: in Model B task 2, switching from post_padding (reference) to pre_padding in stack_conv (reference) implies a decrease in performance.

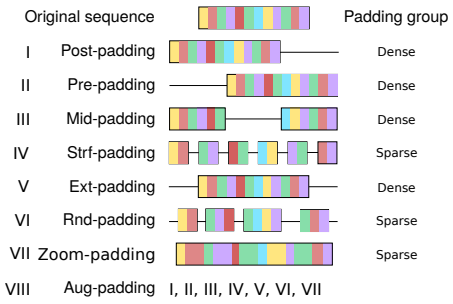

Figure 31: Different types of padding analysed in this study. Each color bin represents an amino acid. Black lines represents zero padding.

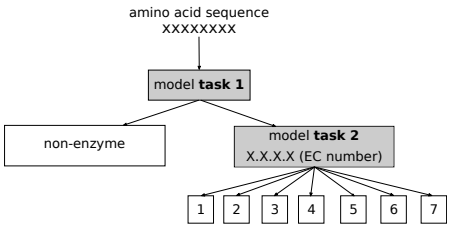

Figure 32: Hierarchical enzyme classification with two levels. Task 1 classifies enzyme/non-enzyme and if enzyme, task 2 classifies by the first digit of the EC number.

6 DATA IMBALANCE

BALANCING DATA ON DEEP LEARNING-BASED PROTEOCHEMOMETRIC ACTIVITY CLASSIFICATION

In silico analysis of biological activity data has become an essential technique in pharmaceutical development. Specifically, the so-called proteochemometric models aim to share information between targets in machine learning ligand-target activity prediction models. However, bioactivity datasets used in proteochemometrics modeling are usually imbalanced, which could potentially affect the performance of the models. In this work, we explored the effect of different balancing strategies in deep learning proteochemometric target-compound activity classification models while controlling for the compound series bias through clustering. These strategies were: (1) *no resampling*, (2) *resampling after clustering*, (3) *resampling before clustering* and (4) *semi resampling*. These schemas were evaluated in kinases, GPCRs, nuclear receptors and proteases from BindingDB. We observed that the predicted proportion of positives was driven by the actual data balance in the test set. Additionally, it was confirmed that data balance had an impact on the performance estimates of the proteochemometrics model. We recommend a combination of data augmentation and clustering in the training set (*semi resampling*) in order to mitigate the data imbalance effect in a realistic scenario. The code of this analysis is publicly available at `https://github.com/b2slab/imbalance_pcm_benchmark`.

6.1 INTRODUCTION

The discovery, design and bring-to-market of a novel small-molecule drug is a very challenging process, and very expensive in terms of money, time and effort (DiMasi et al., 2016). Computer-Assisted Drug Design (CADD) methods can help to improve and refine the identification of hits in the first steps of drug development, thus having a huge positive impact on the costs of the whole process(Qiu et al., 2017). Traditionally, interactions between ligands and targets have been predicted in CADD through a Quantitative Structure-Activity Relationship (QSAR) approach (Hansch and Fujita, 1964). In QSAR, a target is fixed and only information from compounds is used for modeling and predicting binding for said target. However, the compartmentalized nature of QSAR does not allow for discovering new cross-interactions

This chapter is a postprint of the following journal article: Angela Lopez-del Rio, Sergio-Picart Armada and Alexandre Perera-Lluna. "Balancing Data on Deep Learning-based Proteochemometric Activity Classification". *Journal of Chemical Information and Modeling, 61(4), 1657-1669 (2021)*

between ligand and targets for which no training data is available (Qiu et al., 2017). Proteochemometrics modeling (PCM) is an extension of QSAR which overcomes this drawback by combining information of both ligand and protein descriptors on a supervised prediction model. PCM allows for the integration of different sources of data in one model and for the general prediction of which ligands will bind to which targets (Bongers et al., 2020).

Both PCM and QSAR usally apply machine learning (ML) techniques such as random forests, support vector machine, logistic regression or partial least squares (Bongers et al., 2020; Qiu et al., 2017). Following the trends in other fields and the growing availability of data, deep learning (DL) has also been increasingly and succesfully applied on bioactivity prediction (H. Chen et al., 2018), specially on QSAR modeling (Ghasemi et al., 2018). The application of DL to PCM followed, taking advantage of public databases (Lenselink et al., 2017; Lopez-Del Rio et al., 2019; Zakharov et al., 2019) and improving the descriptor representation (Jaeger et al., 2018; Jimenez et al., 2018).

However, an important issue for PCM and QSAR DL models is the amount and quality of data when compared to other fields of application, since increasing the number of data samples in drug discovery is expensive and thus, often infeasible (X. Yang et al., 2019). This poses a problem, since neural networks require a large quantity of training data in order to actually learn. While in other fields this problem is alleviated through data augmentation, i.e. an artificial increase of the number of observations of the training set to help the model generalize, this regularization technique is not yet commonly used in CADD. Some studies have considered different variants of the SMILES of each molecule as a way of data augmentation (Bjerrum, 2017; Kimber et al., 2018), but despite its proven benefits, its use is not widespread yet. This is partly due to the lack of consensus in the input representations, where alternatives to SMILES are often used.

Another factor highly affecting QSAR and PCM models is data imbalance, since the class definitions based on bioactivity data can result in highly skewed labels. In this regard, Zakharov et al (Zakharov et al., 2014) explored how data balancing affected self-consistent regression QSAR models using highly imbalanced PubChem bioassays. The study proposed a method including cost-sensitive learning and under-sampling approaches to obtain more accurate predictions. Using the same data, Korkmaz explored how data balancing affected DL-based QSAR models (Korkmaz, 2020). The study concluded that imbalance has indeed a negative impact on the performance of the models, but that this impact could be alleviated by applying oversampling methods like SMOTE (Synthetic Minority Oversampling Technique) (Chawla et al., 2002) on the fingeprint representations of the molecules. Besides, oversampling methods could also serve the purpose of augmenting the original dataset.

While the effect of data imbalance on model performance has been studied for shallow ML and DL QSAR, up to our knowledge, there are not analogous studies yet for PCM. In PCM, modeling information between targets is shared, which may compensate those for which activity data is very imbalanced. However, it is still to be proved if this compensation does happen or if the results are actually dominated by the original imbalance of each target.

Recently it has been shown that for the validation of PCM models, it is important to control the chemical series bias through clustering techniques in order to get more reliable performance estimates (Lopez-Del Rio et al., 2019; Mayr et al., 2018). This adds a complexity layer to the imbalance handling, since clustering can affect the data balance in PCM. Since Korkmaz and Zakharov et al did not consider the potential similarity between different compounds when validating their results (Korkmaz, 2020; Zakharov et al., 2014), its impact on data balancing is yet to be tested.

In this paper, we study the effect of different balancing strategies in DL-based PCM target-compound activity classification models. While handling data imbalance, we also study how to integrate the compounds clustering in this process. We describe the behavior of model predictions and performance according to imbalance handling.

6.2 MATERIALS AND METHODS

6.2.1 Data

We evaluated the different balancing models on the benchmark dataset used in DeepAffinity (Karimi et al., 2019). The original dataset contains binding data from BindingDB (Gilson et al., 2016), merged with the amino acid sequence information from UniRef (Suzek et al., 2014) and the SMILES representation of compounds from STITCH (Kuhn et al., 2007). The original dataset consisted of IC50, K_i or K_d values from 829,033 compound-protein pairs. We classified the dataset proteins into the main protein families according to the release 2018_09 from Uniprot (Consortium, 2018) and focused our study on proteins of the kinase family. Our results were further validated on the G protein-coupled receptors (GPCR), nuclear receptors (NR) and proteases (PR) families (separately). Binding activities were in logarithm form, so a threshold of 6 was applied in order to have binary labels for classification (active/inactive). Table 12 summarizes the final dataset we used in our analysis. The same descriptive table, but for GPCR, NR and PR families, can be seen in Table S29 of the Supporting Information.

Entity	Number
Compounds	84,643
Targets	490
Ligand-target pairs	129,997
Actives	99,158
Inactives	30,839

Table 12: Summary of the kinases subdataset.

In Figures S72 and S73 from the Supporting Information, the proportion of actives/inactives for each protein of each of the studied protein families is represented in more detail.

6.2.2 Descriptors

We represented compounds by their molecular fingerprints, in which structural information is represented by bits in a bit string. We used the fingerprints from PubChem (Kim et al., 2019) provided in DeepAffinity (Karimi et al., 2019). In these, basic substructures of compounds are encoded in a 1D binary vector with a length of 881 bits.

We represented proteins by raw amino acid sequences transformed to one-hot encoding. Each amino acid was represented by a binary vector of length 26. Protein sequences were then normalized to the maximum length of 1499. Those sequences shorter than 1499 were zero-padded. According to the recommendation of our previous work (Lopez-Del Rio et al., 2020), we tuned the padding type and obtained the best results with pre-padding (adding zeros to the beginning of the sequence).

6.2.3 Validation strategy

A splitting strategy based on compound clustering (both of actives and inactives) was applied to the bioactivity data, omitting target information. Clustering-based validation strategies have been used to avoid the compound series bias, making sure that there are no similar molecules both in training, validation and test sets (Mayr et al., 2018; Ramsundar et al., 2017; Rodriguez-Perez and Bajorath, 2019). We followed the implementation of our previous study on cross-validation strategies in PCM (Lopez-Del Rio et al., 2019), where K-means clustering with $k = 100$ was applied to the fingerprint description of the compounds. Data was divided in training, validation (for selecting the best epoch) and test (for evaluating the performance) sets with a proportion of 80/10/10%. This splitting was randomly performed 10 times (folds) in order to test the consistency of the results, thus training and testing each model in 10 different data partitions. As further explained in the next subsection, for some balancing strategies the clustering was applied before the resampling and for others it was applied afterwards.

6.2.4 Balancing strategies

We chose an oversampling method to balance data since oversampling was shown to improve performance in the Korkmaz study of data imbalance in DL-based QSAR (Korkmaz, 2020) and in a systematic study of data imbalance with CNNs (Buda et al., 2018). Oversampling methods increase the number of samples in the minority class to create a balanced data set. Specifically, we used the SMOTE oversampling technique (Chawla et al., 2002), which creates synthetic data points of the minority class similar to those available. Resampling with SMOTE was done in a per protein basis, so that each protein would be balanced. Some proteins had to be discarded in certain strategies, since there were either only active or inactive ligands, or the number of samples in the minority class was smaller than the number of neighbors used for constructing the synthetic samples ($k = 5$) and SMOTE was not applicable.

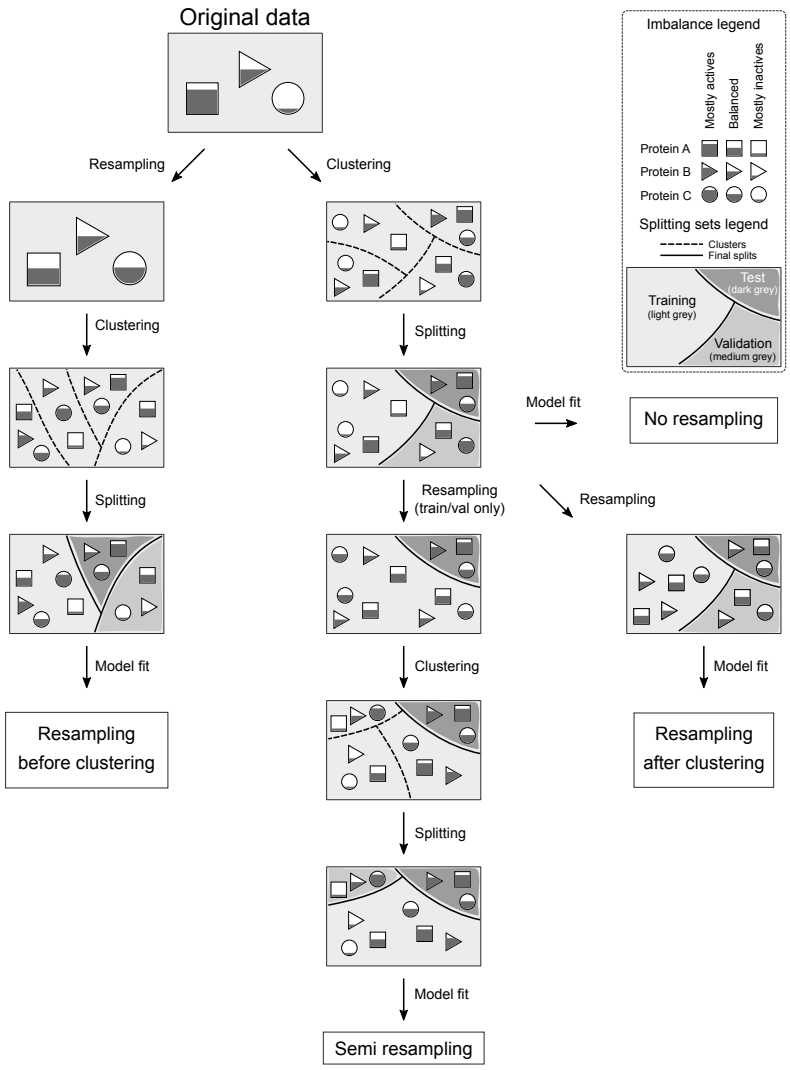

Figure 33: Description of the four balancing strategies that were applied to the bioactivity data. ***Resampling before clustering***, where resampling per protein is applied prior to clustering and splitting; ***resampling after clustering***, where data is first clustered and split and then each protein activity data in each set is resampled; ***semi resampling***, in which the splitting is performed and then the test set is kept without resampling but the training+validation set is resampled and clustered; and ***no resampling***, in which the imbalance of the original data is kept and clustering is applied prior to splitting. Dashed lines indicate clusters and solid lines delineate the final splits; in the latter, training, validation and test sets can be recognised by their shade intensity. Filled shapes illustrate the active ratios for each of the three example proteins, in every set or cluster.

Unlike Korkmaz, that applied data balancing methods to each training set (Korkmaz, 2020), we tested four different combinations of balancing, data clustering and splitting (see Figure 33): ***no resampling***, in which bioactivity data for each protein was taken as it was, and clustering was applied in order to perform the splitting; ***resampling after clustering***, in which after clustering data and splitting it into training, validation and test, each protein activity data in each set was resampled and attained a 50% actives/inactives

proportion; ***resampling before clustering***, in which, opposite to the previous strategy, resampling was applied prior to clustering and splitting, so while the global protein-wise proportion of actives/inactives was 50%, it did not have to be 50% within each splitting set; and ***semi resampling***, in which the splitting performed in the *no resampling* strategy was reused, the test set was kept without resampling but the training+validation set was resampled, re-clustered and re-split into train and validation.

The overall amount of resampling rounds, by strategy, was: 0 in *no resampling*, 30 in *resampling after clustering* (in each fold, one resampling in the training set, one in the validation set and one in the test set), 1 in *resampling before clustering* (from which the 10 folds were calculated), and 10 in *semi resampling* (in each fold, one resampling in the training and validation sets combined). In turn, each resampling round consisted of oversampling each one of the available proteins in the corresponding set. The total number of active and inactive protein-compound pairs in each strategy, splitting set and protein family can be seen in Table S30 of the Supporting Information.

6.2.5 Prediction models

We built a DL model for studying the impact of different data balancing strategies in state-of-the-art PCM. Besides, a random prediction was generated to have an absolute, input-naïve baseline. Having predictions from a random baseline served two purposes: characterising how well random predictions scored in each performance metric in a scenario with varying data imbalance, and putting the performance of DL models in context.

Random baseline

A random baseline was computed according to the actives/inactives ratio of the training set for each strategy and each fold. Let f be the fraction of actives in the training samples involving a protein, and n the number of samples to be predicted in the test set for that protein. The random baseline is obtained by first sampling $\lfloor fn + 0.5 \rfloor$ values from a uniform distribution in $[0.5, 1]$ (actives) and $n - \lfloor fn + 0.5 \rfloor$ values from a uniform distribution in $[0, 0.5]$ (inactives), then concatenating both and shuffling. This procedure keeps the active/inactive balance by design while producing random activity predictions.

Deep learning model

We studied the impact of data balancing strategies on a DL model. We followed the Korkmaz strategy of selecting a simple, well-established architecture whose complexity issues would not be a confounder of the factor under study (Korkmaz, 2020). We refrained from using Long Short-Term Memory networks since they have convergence issues when training sequences longer than 1000 elements (S. Li et al., 2018). Model hyperparameters were tuned using the validation set, choosing the simplest working architecture. As in our previous work (Lopez-Del Rio et al., 2019), the DL PCM model consisted of two analysis blocks. The amino acid sequence analysis block was a 1D

convolutional neural network. The fingerprints analysis block consisted of a feed-forward neural network. Dropout was used in both branches to prevent overfitting (N. Srivastava et al., 2014). The representations built by the compound and target analysis blocks were then merged and the information was passed through a softmax activation unit, which quantified the ligand-target pair activity probability. A schematic representation of the DL-based PCM model can be found in Figure S74 of the Supporting Information, along with further details on the optimised hyperparameters. In the training process, the weights of the selected model were those from the epoch with the maximum accuracy (proportion of correct predictions) on the validation set. This process was run for each strategy and fold. Then, each selected model was used to predict on their corresponding test set.

6.2.6 Characterization of data balance

The data balancing strategy had an impact on the actual data balance, defined as the proportion of active molecules for a protein.

Data balance (protein) = Proportion of actives (protein) = $\frac{n\_active\_compounds}{n\_total\_compounds}$

Thus, a comprehensive analysis of data balance was carried to better understand and interpret performance results. For each of the balancing strategies, the original distribution of active ratios per protein was characterized. We also compared the original imbalance of the training and test sets for each strategy to explore possible trends, and studied the effect that other covariates (the protein length and the number of interactions of each protein in its corresponding set and fold) might have on the original test set imbalance.

The next key question was to narrow down the factor driving the proportion of actives in the predicted data (as opposed to the original data). The main options under consideration were: (1) a constant, global imbalance that the model would learn from the whole dataset; (2) the protein-wise imbalance that the model would learn in the training set and (3) a test set-driven imbalance, based on its actual imbalance. To answer this, the test set predictions were binarized with a probability threshold of 0.5 and the proportion of predicted actives was computed by protein and also compared to the ratios of the original test and training sets.

6.2.7 Performance metrics

The resampling strategies were assessed with various performance metrics for binary classifiers and prioritisers. The selection was based on those used by Korkmaz (Korkmaz, 2020): balanced accuracy, F1-score, Matthews correlation coefficient (MCC) and area under the ROC curve (AUROC). All of them are insensitive to class imbalance. In the case of F1-score, we used the macro-average, which is computed by averaging the F1-score for the active and inactive labels. Further details on the definition of these metrics can be found in the Supporting Information.

The performance metrics were computed on the predictions of each selected model in its corresponding test set. For each combination of resampling strategy, fold and protein, we computed the performance of (1) the

random baseline, and (2) the DL model. AUROC was computed from raw predicted probabilities, while F1-score, balanced accuracy and MCC were derived from the binarized predictions. We tested the significance of the differences between strategies by means of nonparametric two-sided Wilcoxon test for paired samples (Wilcoxon, 1945).

6.2.8 Explanatory models

Performance metrics and predicted ratios were further described through linear models built upon the different combination of variables considered in this analysis. Our prior work in similar scopes had found them insightful, since they allow for a statistical analysis of the contribution of each factor under study (Lopez-Del Rio et al., 2020, 2019; Picart-Armada et al., 2019). Each of the data points used for fitting a explanatory linear model corresponded to a different protein. Simpler claims were investigated with Pearson's r for linear correlation, using confidence intervals (CI) and p-values for significance.

On the one hand, the predicted ratio of actives (r_{pred}) was modelled through the quasibinomial logistic model (Hardin et al., 2007) in equation 14, stratified by strategy, in order to quantify the effect of different variables of interest.

$$r_{pred} \sim r_{training} + r_{test} + log10(n_{int}) + log10(n_{seq}) + k_{fold} \tag{14}$$

Specifically, the main variables of interest in this model were the actual ratios in the training ($r_{training}$) and in the test (r_{test}) sets, both numeric between 0 and 1. As additional covariates, the number of interactions (n_{int}) and the sequence length (n_{seq}) (both numerical) and the fold number (k_{fold}, categorical) were also included. This model was not computed for the *resampling after clustering* strategy, since the data balance (and thus, the predicted active ratio) is enforced.

On the other hand, each performance metric was explained through the linear model described by the Equation 15.

$$metric \sim strategy + log10(n_{int}) + log10(n_{seq}) + k_{fold} \tag{15}$$

The response was the quantitative metric of interest in each case (one model per metric), while strategy was categorical (*no resampling*, *resampling after clustering*, *resampling before clustering*, *semi resampling*). The same covariates as in Equation 14 were added.

However, before evaluating the DL model, the performance metrics of the baseline were characterised: the strategy variable was tested with a type 3 analysis of variance (ANOVA) (Fisher, 1992) in order to pinpoint the imbalance-sensitive and insensitive metrics. Metrics were called imbalance-sensitive if the imbalance-aware random baseline exhibited different performances between resampling strategies.

The imbalance insensitive metric models were fitted analogously to the baseline performance models (with Equation 15). However, to address the pitfalls of the direct comparison of metrics whose baselines might differ,

imbalance sensitive performance metrics were defined and modelled as follows:

$$adj\_metric = metric - baseline \tag{16}$$

And thus, adjusted performance metrics were also described with the Equation 15 but changing the response to adj_metric of Equation 16:

$$adj\_metric \sim strategy + log10(n_{int}) + log10(n_{seq}) + k_{fold} \tag{17}$$

Note that while all the metrics but MCC were non-negative, the adjusted metrics could show negative values when the performance of the DL model was lower than that of the baseline.

Reference categories for categorical variables were *no resampling* for strategy and 0 for fold. Each term of the fitted model represents the difference between its specified category and the reference category of that variable.

6.2.9 Implementation

We trained every model with an Adam optimizer (Diederik P Kingma and Ba, 2014) (learning rate= 5×10^{-4}, $\beta_1 = 0.1$, $\beta_2 = 0.001$, $\epsilon = 1 \times 10^{-8}$ and decay rate defined as the learning rate/number of epochs) for 100 epochs, with a batch size of 128 both for training and validation. Models were implemented in Python 3.6.9 (Keras (Chollet et al., 2015) 2.3.1 using Tensorflow (Martín Abadi et al., 2015) 2.1.0 as backend) and run on two NVIDIA GeForce GTX 1070 GPUs. SMOTE data balancing was applied using the imbalanced-learn Python package (Lemaître et al., 2017). The statistical processing of results was performed in R software (3.6.3) (R Core Team, 2015).

6.3 RESULTS

Unless stated otherwise, the results showed in this section refer to the kinases protein family.

6.3.1 Characterization of the original data balance

Distribution of the actives ratio

Figure 34 displays the original distribution of the actives ratio in the training and test sets. Test sets tended to magnify data imbalance, creating around 24% of the times extreme cases, i.e. all actives or all inactives, not present in the training set. Strategy-wise, *no resampling* kept similar data distributions in training and test; *resampling before clustering* and *semi resampling* led to a more balanced training set, but an imbalanced test set, and *resampling after clustering* only kept totally balanced proteins in both training and test sets.

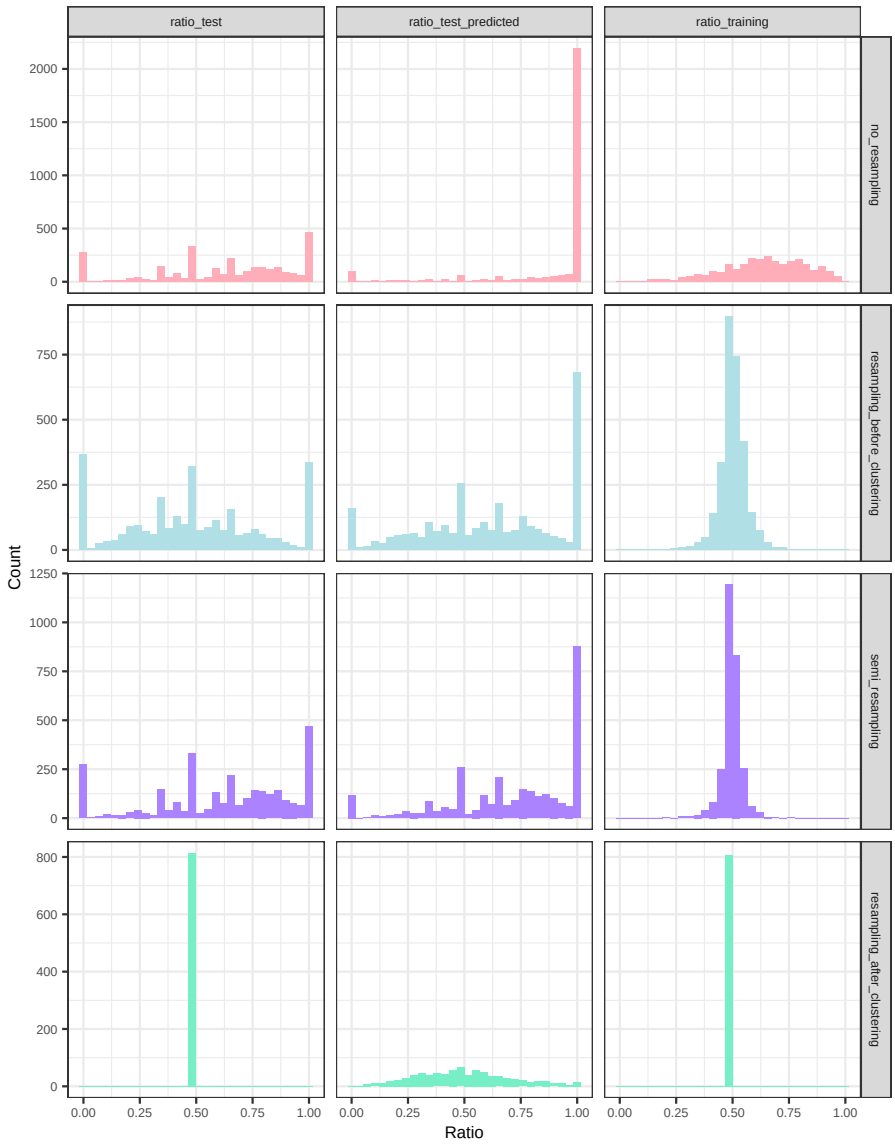

Figure 34: Histograms of the active ratios in the training set, and in the test set (both original and predicted by the deep learning model), within each resampling strategy. Each histogram combines all the folds.

Training and test imbalance comparison

Figure 35 revealed both positive, negative and null trends between the training and test protein balances, and Table S32 of the Supporting Information quantifies these correlations. *No resampling* showed a positive correlation between both (Pearson's r 95% CI: $[0.338, 0.400]$, $p < 10^{-16}$), i.e. proteins were prone to keep their (im)balance in training and test sets. *Resampling before clustering* showed an inverse relationship (Pearson's r 95% CI: $[-0.457, -0.398]$, $p < 10^{-16}$), which was expected since this strategy started from globally balanced proteins and after the clustering, an imbalance in one direction in the training set entailed an inverse imbalance in the test set. *Semi resampling* led to uncorrelated train and test balances (Pearson's r 95% CI: $[-0.024, 0.051]$, $p = 0.48$), expected since the training set was resampled,

breaking any correlation with the test set balance. *Resampling after clustering* always kept balanced proteins, by design.

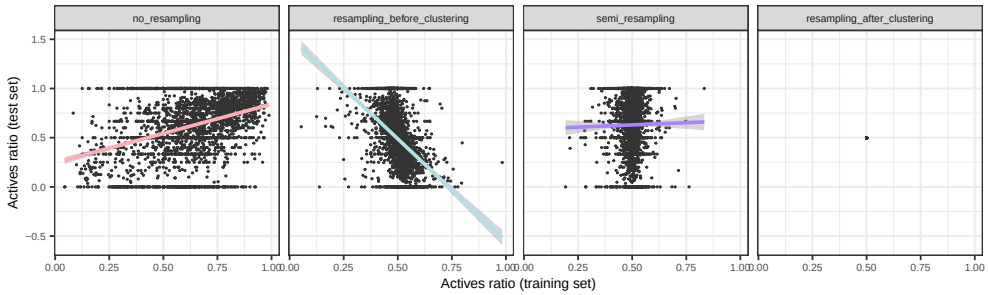

Figure 35: Comparison of the training and test original active ratios, by resampling strategy. Linear fit trends were added by strategy, and the shadowed areas indicated the 95% CI of the expected value. Each plot combines all the folds.

Other covariates

The effect that the number of interactions for each protein in its corresponding set and fold, and the protein length (i.e. number of amino acids) had on the test set imbalance was investigated (Figures S76-S77 and Tables S33-S34 of the Supporting Information). Proteins with greatest imbalance tended to be among those with the least interactions (Table S33: Pearson's r 95% CI $[-0.097, -0.026]$, $p = 8.01 \cdot 10^{-4}$ for *no resampling* and *semi resampling*; $[-0.307, -0.240]$, $p < 10^{-16}$ for *resampling before clustering*). The sequence length had no consistent effect on the protein imbalance (Table S34: Pearson's r 95% CI $[-0.052, 0.020]$, $p = 0.37$ for *no resampling* and *semi resampling*; $[-0.082, -0.009]$, $p = 0.014$ for *resampling before clustering*).

6.3.2 Analysis of the predicted proportions of active compounds

Figure 34 represents the ratio of predicted actives by protein and Table S35 of the Supporting Information summarizes the percentage of proteins with all actives or inactives (extreme cases). They show that *no resampling* strategy was inclined to predict everything as positives (71.6% of the time, compared to 3.5% for predicting all negatives). *Resampling before clustering* and *semi resampling* alleviated the imbalance in the predictions, but still retained a spike of proteins where all the compounds were predicted as positives (23.4% and 29.1%) and negatives (5.5% and 4%). *Resampling after clustering* kept a wide and symmetric distribution of predicted actives, with only 1.2% predicted as all actives and 0% as all inactives.

Figure 34 also puts the ratio of predicted actives in context with the original training and test ratios: the distribution was more similar to that of the test proportions than to that of the training ones (except *resampling after clustering*, since those proportions were constant).

Figure 36 puts the predicted ratios in context of the training ratios and Table S36 of the Supporting Information quantifies their correlations, elucidating a variety of trends: (1) *no resampling* shows a positive trend be-

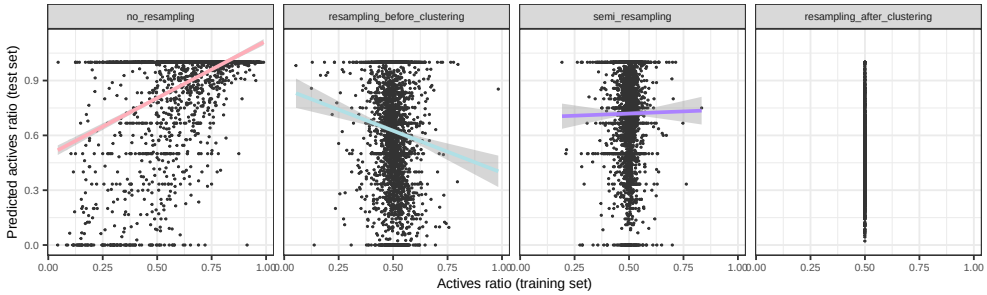

Figure 36: Predicted ratios, as a function of training ratios, by resampling strategy. Linear fit trends were added by strategy, and the shadowed areas indicated the 95% CI of the expected value. Each plot combines all the folds.

tween the training and the predicted ratio (Pearson's r 95% CI: $[0.440, 0.496]$, $p < 10^{-16}$), but since the training and the test ratio are also positively correlated (Figure 35), the latter could be the one driving the predicted ratio of positives; (2) *resampling after clustering* had a constant training ratio, meaning that the predicted ratio was not explainable by differences in training ratios; (3) *resampling before clustering* showed instead a negative relation between the training and the predicted ratio (Pearson's r 95% CI: $[-0.130, -0.058]$, $p = 3.77 \cdot 10^{-7}$), but since the former and the test ratio also anticorrelated (Figure 35), the simplest explanation was that the test ratio drove the predicted test ratio; (4) *semi resampling* showed no apparent correlation between the predicted ratio and the training ratio (Pearson's r 95% CI: $[-0.029, 0.045]$, $p = 0.68$).

The models in Equation 14 that describe the predicted ratio of actives for each balancing strategy are summarized in Tables S37-S38 of the Supporting Information. For *semi resampling* and *resampling before clustering* (Table S37), the original actives ratio in the test set had a positive, significant effect on the predicted actives ratio ($\beta = 0.945$ and 0.784, both $p < 10^{-16}$). However, the original actives ratio of the training set showed no evidence of affecting the predicted ratio ($\beta = 0.197$ and -0.446, $p = 0.73$ and 0.31). Conversely, for the *no resampling* strategy (Table S38), both the original training ($\beta = 8.312$, $p < 10^{-16}$) and test ratios ($\beta = 1.102$, $p = 2.6 \cdot 10^{-9}$) had positive, significant effects on the predicted actives ratio. In the three models, the number of interactions per protein had a significant, negative effect ($\beta = -0.391$, -0.396 and -1.24, all $p < 10^{-16}$), and some of the folds entailed significant variations of the predicted ratio.

6.3.3 Performance metrics

Baseline performance

Figure 37 shows a fold-averaged picture of the metrics by protein and by model type (DL or input-naïve baseline). Visual inspection suggested that the F1-score, accuracy, and possibly balanced accuracy were affected by the baseline data imbalance. To quantify this finding, the model in Equation 15 was fitted to the baseline performance metrics. According to Table S40 of the

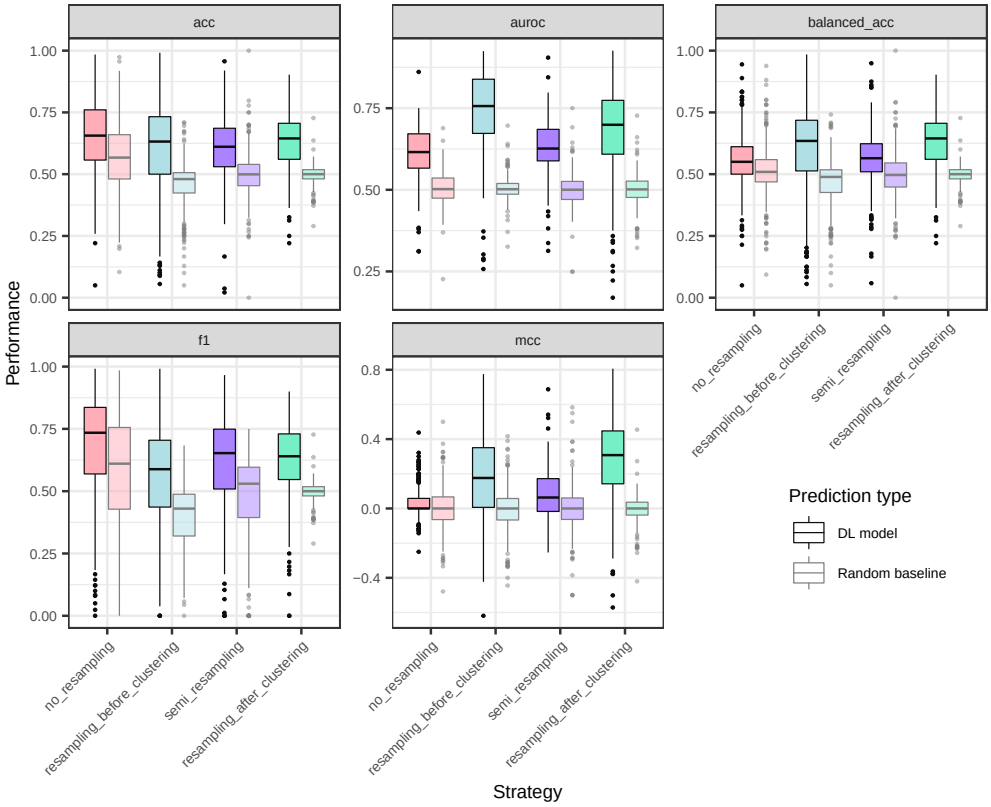

Figure 37: Absolute performance metrics for balancing strategies and their corresponding imbalance-aware random baselines. Data points correspond to proteins, averaged over folds.

Supporting Information, the strategy term was significant (type 3 ANOVA, $p < 10^{-16}$, $p < 10^{-16}$ and $5.61 \cdot 10^{-11}$) for those three metrics, and non-significant in AUROC and MCC ($p = 0.91$ and 0.82). Based on this, metrics were divided in two types: (1) imbalance-sensitive, if the baseline was different between strategies, and (2) imbalance-insensitive, if the baseline was constant.

Deep learning model

Figure 37 displays an overview of fold-averaged performances, where strategies are paired with their baselines. Undefined metrics in edge cases were excluded. This mainly affected AUROC, where the number of proteins with metrics dropped around 25% for *semi resampling*, *resampling before clustering* and *no resampling* (Table S41 of the Supporting Information). Figure 37 brought the dilemma of direct strategy comparison with imbalance-sensitive metrics, which was especially apparent for the F1-score and its high baseline in *no resampling* (quartiles: Q1 = 0.428, median of 0.611, Q3 = 0.756, Table S39 of the Supporting Information).

ABSOLUTE, BASELINE-NAÏVE PERFORMANCE Absolute metric models (not accounting for baselines) were fitted following Equation 15, analogously to the baseline performance models. The strategy term would always explain variance (type 3 ANOVA, p-values ranged between $2.89 \cdot 10^{-15}$ and

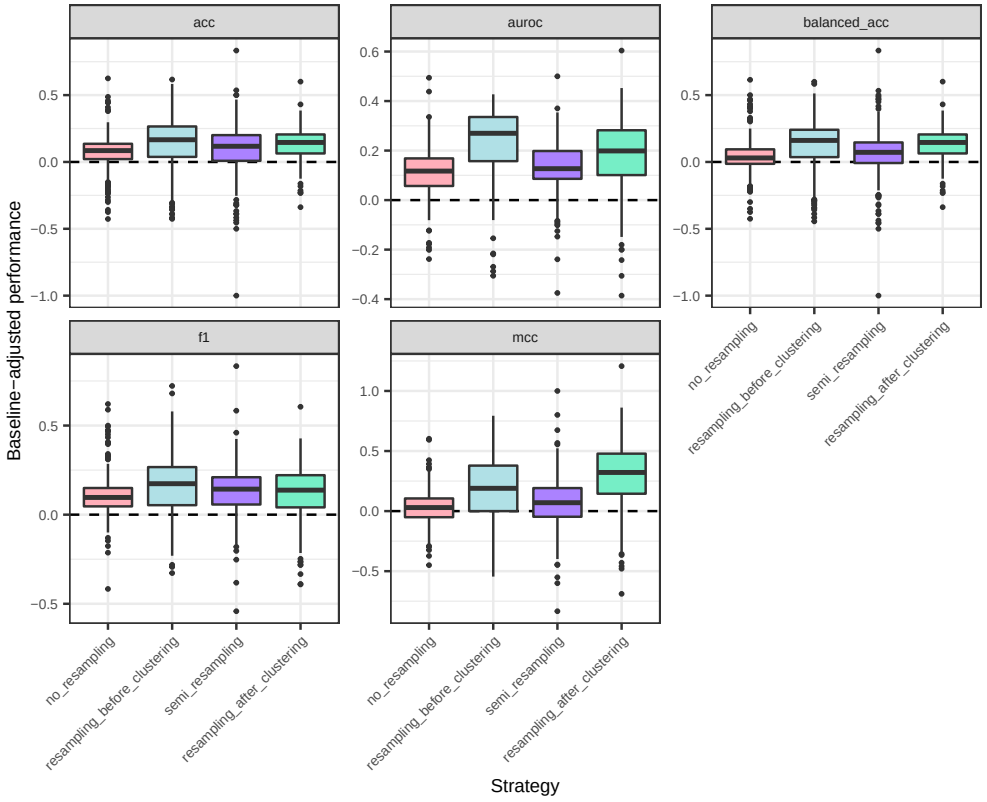

Figure 38: Baseline-adjusted performance metrics for balancing strategies. Data points correspond to proteins, averaged over folds. Values are positive when the DL model performs better than its paired imbalance-aware baseline, and negative otherwise.

$p < 10^{-16}$, see Table S42 in the Supporting Information). The models showed different behaviour in imbalance-sensitive and insensitive metrics (Table S43 of the Supporting Information). Pairwise comparisons of the strategy term coefficients using Tukey's method would point to two apparently conflicting scenarios (Figure S81 of the Supporting Information), further confirmed when prioritizing the strategies according to their expected performance through the linear models (Figure 39 and Table S44 of the Supporting Information): (a) *no resampling* was suggested the best strategy by accuracy and F1-score (95% CI of expected performances: $[0.701, 0.723]$ and $[0.754, 0.779]$), but this was confounded by the fact that it also held the highest baselines, and (b) *resampling before clustering* and *resampling after clustering* kept the highest performance estimates in AUROC (95% CI $[0.699, 0.724]$ and $[0.670, 0.708]$), MCC (95% CI $[0.244, 0.268]$ and $[0.296, 0.337]$) and balanced accuracy (95% CI $[0.619, 0.640]$ and $[0.634, 0.670]$).

BASELINE-ADJUSTED PERFORMANCE A descriptive plot of the adjusted metrics (Figure 38) pointed to a different scenario than that of the absolute ones (Figure 37).

Again, the strategy term was always significant (type 3 ANOVA, p-values ranged between $2.78 \cdot 10^{-9}$ and $p < 10^{-16}$, Table S45 of the Supporting Information). Baseline adjustment brought a unified behaviour across the models (Table S46 of the Supporting Information), further confirmed in pairwise

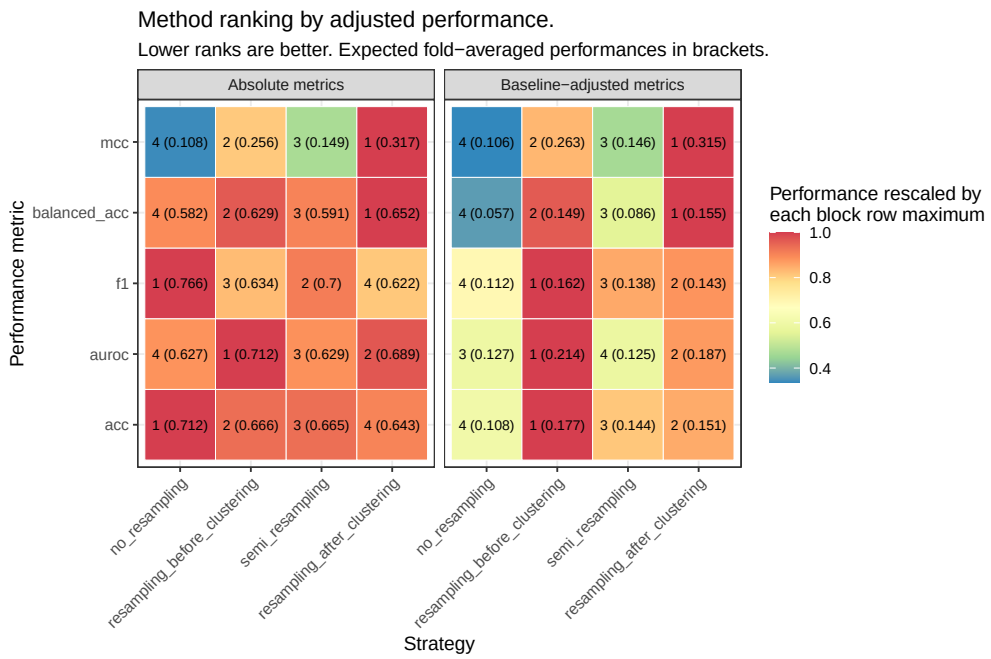

Figure 39: Resampling strategy ranking according to their absolute (left block) and baseline-adjusted performances (right block), estimated through the corresponding linear model of each metric. For baseline-adjusted metrics, only the improvement over the baseline is displayed. The ranking, ranging from 1 (best) to 4 (worst) in each row and block, was based on the expected performance, averaged over folds and indicated in parentheses. The colour scale varies between the block row-wise maximum (red) and 0 (blue).

coefficient comparison (Tukey's method, Figure S81 of the Supporting Information) and in their expected performance (Figure 39 and Table S47 of the Supporting Information): *resampling before clustering* and *resampling after clustering* had the highest performance estimates (expected improvements over baseline ranging from 0.149 to 0.263 and from 0.143 to 0.315 in all metrics), followed by *semi resampling* (0.086 to 0.146) and finally by *no resampling* (0.057 to 0.127).

6.3.4 Further validation with other protein families

We repeated all the previous analysis on three protein families to confirm whether the claims obtained for the kinases protein family could be generalized to other families. Those families were G protein-coupled receptors (GPCR), nuclear receptors (NR) and proteases (PR). Supplements 3, 4 and 5 gather with detail the replication of the kinases analysis in GPCRs, NRs and PRs. In general, the main observations and recommendations hold in GPCRs, NRs and PRs.

Compared with kinases, GPCRs contained almost 60% more protein-compound pairs, PRs were roughly even, and NRs had about 20% of their interactions. GPCRs were more imbalanced towards the actives than kinases while NRs and PRs kept more balanced active ratio distributions (Figure S72 of the Supporting Information).

The distributions of active ratios and the comparison between training and test set imbalances hold in GPCRs, NRs and PRs. The only exception was *semi resampling* in GPCRs, exhibiting a slight positive correlation (Pearson's r 95% CI $[0.029, 0.105]$, $p = 5.91 \cdot 10^{-4}$) between training and test balances (see Table S48 of the Supporting Information) instead of no correlation. The effect of the number of interactions and the sequence length on the protein imbalance were also replicated on the GPCRs, NRs and PRs.

Kinases, GPCRs, NRs and PRs mainly agreed on the predicted active proportions analyses, except for the n_interactions coefficient, non-significant in the *semi resampling* strategy in GPCRs, NRs and PRs. Still, the *semi resampling* model was always the clearest scenario to show that the predicted active proportions were driven by the actual proportions in the test set, rather than those in the training set.

Regarding performance, the explanatory linear models on GPCRs, NRs and PRs also identified accuracy, F1 and balanced accuracy as sensitive to data imbalance.

The analysis of absolute metrics in GPCRs was analogous to that of kinases, while NRs and PRs showed some differences. Those mainly involved which metrics place *no resampling* as the best strategy (accuracy, balanced accuracy and F1-score for NRs; and also AUROC in PRs), and *resampling after clustering* not being suggested as the best strategy anymore.

As for adjusted performances, GPCRs showed essentially the same facts as kinases. In NRs and PRs, *resampling before clustering* still showed the best performance in general, but *resampling after clustering* lost its shared dominance with the former. This implied that augmenting the test set was not the largest performance drive anymore, which might be explained by the

more moderate data imbalance in NRs and PRs as compared to kinases and GPCRs. On the other hand, our main recommendation remained unchanged, since *semi resampling* still compared favourably to *no resampling*, with less significant changes (especially in NRs with their sensibly reduced sample size), but always in favour of the former if present.

6.4 DISCUSSION

The impact of clustering in final imbalance was strategy-dependent

This study is focused on the characterization of the data imbalance present in bioactivity datasets, as well as how to address it. Bioactivity data also poses the problem of chemical series, i.e. sets of similar molecules with similar activities, that result in inflated performance metrics when split between training and test sets. We addressed those via a clustering prior to the splitting, ensuring that similar molecules would belong to the same set.

The first observation was that clustering modified data imbalance in a strategy-dependent way. When the starting set was perfectly balanced (strategy *resampling before clustering*), clustering and splitting induced a degree of imbalance, particularly visible in the heavier tails of the active ratios distributions in the test set. Compared to training, the lower sample sizes in the test set may also cause extreme imbalances more often. On the other end, this effect was only moderate in *no resampling*, where the distribution of actives ratio was similar in train and test, but that of test had more extreme proteins with either all actives or all inactives.

Besides the overall changes in data imbalance, strategies differed in how the imbalance of a certain protein in the training set would translate to the test set. The positive trend in *no resampling* suggests that existing data imbalances tended to persist after the clustering and splitting. The negative trend in *resampling before clustering* hints that, in the absence of imbalance, clustering will induce it. The flat trend in *semi resampling* supports that the imbalance induced with the clustering in the training set, which was balanced with SMOTE beforehand, is independent from the original imbalance in the dataset (present in the test set).

The predicted active proportion was driven by the test set rather than the training

The original distribution of actives ratio in each of the balancing strategies affected the predicted ratio of actives by the models. Due to the lack of correlation between training and test ratios (Figure 35), the *semi resampling* strategy was the ideal scenario to disentangle their effect on the predicted ratio of actives (see model in table S32 of the Supporting Information). Its additive model suggested that the original ratio of actives in test explained the predicted proportions, rather than the training ratio. We also found that the number of interactions per protein was a relevant factor: the more

interactions, the less active proportion, suggesting that the extreme cases with all predicted as actives tended to be proteins with few interactions.

Likewise, *resampling before clustering* showed negative correlation between training and test ratios, also providing a reasonably good scenario to distinguish their effects (Table S32 from the Supporting Information). Its explanatory model confirmed both conclusions from the model in the *semi resampling* strategy, with similar estimates (Table S37).

The explanatory model for the *no resampling* strategy (Table S38 of the Supporting Information) suffered from the positive correlation between training and test ratios, which could be confounded. Both original training and test ratios showed a positive effect on the predicted fraction of actives. Although the estimate was larger and more significant for the training ratio coefficient, the confounding effect and the very skewed distribution of the predicted ratios deemed this model inconclusive.

Imbalance-sensitive metrics required baseline adjustment

The prediction task studied here posed a particular challenge: data imbalance happened on a protein basis, and the imbalance of certain proteins could be extreme (very low or high), moving away from the global actives ratio. Each resampling strategy would lead to different protein-wise imbalance patterns. The baseline performance of some metrics (accuracy, F1-score and balanced accuracy) was different between strategies, while it was constant for others (AUROC and MCC). The data-driven division into imbalance-sensitive and insensitive metrics was an important step to understand the opposite conclusions reached within each metric type after direct performance comparison between strategies (Figure 39).

The direct comparison of resampling strategies with imbalance-sensitive metrics would be confounded by the imbalance-induced bias in the metrics and the protein-wise imbalance differences between strategies. We found that adjusting by the baseline metrics (see Equation 17) brought an agreement in the conclusions obtained by both imbalance-sensitive and insensitive metrics. In turn, the same conclusions were obtainable by direct comparison of imbalance-insensitive metrics. Because of this, our recommendation is to include imbalance-aware baselines and to adjust imbalance-sensitive metrics when used for model selection.

Augmenting the test set was the largest performance drive

Our results showed that the largest impact in performance estimates was the application of data augmentation to the test set: *resampling before clustering* and *resampling after clustering* tended to outperform *semi resampling* and *no resampling*. However, augmenting the test set might not faithfully reflect new data anymore, and could artificially inflate the performance estimates: models may specialize in discriminating between original and resampled data points instead of actives and inactives. Our validation with other protein families (NRs, PRs) suggested this fact might not apply when the interaction data is more balanced.

Resampling improved performance when keeping the original test set

On the other hand, *semi resampling* outperformed *no resampling* in four out of five metrics (Tukey's method, $p < 0.05$, Figure S81 of the Supporting Information), which supported data augmentation usefulness even if the data balance in the test set differed from that of the training set. This was consistent with the observation that the main influence on the predicted actives ratio in the test set were their actual ratios in the test set instead of the original ratios in the training set. Combined with the less skewed distributions of predicted active ratios of *semi resampling* against *no resampling* (Figure 34), we recommend *semi resampling* for future studies.

Using external protein family datasets for validation suggests replicability of the main guidelines

The results obtained by the kinases and those of the GPCR, NR and PR proteins, used as external validation sets for the model fitting and evaluation, pointed to the same general picture with aligned conclusions. The differences could arise from changes in data imbalance (NRs and PRs were less imbalanced, while GPCRs were more) and number of protein-compound pairs (GPCRs had more interactions, while PRs had less). The variety of scenarios under consideration suggests that the guidelines for proteochemometrics models of our study provide sensible defaults to more protein families.

Similarities with existing literature

In this paper we have confirmed that data balance has an impact in DL proteochemometric target-compound activity models. Zakharov et al and Korkmaz arrived to a similar conclusion in a QSAR setting (Korkmaz, 2020; Zakharov et al., 2014), the latter also using DNN models for classification. More specifically, Korkmaz stated that the higher the imbalance for a protein, the worse the model performance (measured by F1-score and MCC).

These studies got the best performances by controlling data balance by means of undersampling techniques (in the case of Zakharov) and oversampling techniques (in the case of Korkmaz). We chose SMOTE for data balancing, an oversampling technique, since the settings of the Korkmaz study were more aligned with ours and because DL models require a large quantity of training data. Specifically, in four out of five metrics, proteins with more interactions were better predicted (table S46 of the Supporting Information) which was also found in the Korkmaz paper.

Within our resampling strategies, *semi resampling* was the most similar to the balancing process in the Korkmaz study, in which the training and validation sets were oversampled (per protein) while the test set was not.

Dissimilarities with existing literature

Technical differences existed in the descriptors used in the three studies. Zakharov et al used Quantitative Neighborhood of Atoms and biological descriptors, whereas Korkmaz used the PaDEL software. We, on the other hand, used the fingeprints from PubChem. The fact that the overall messages are consistent suggests a degree of independence from the input encoding.

More importantly, Zakharov and Korkmaz studies did not take into account the control of the compound series bias. This step is necessary for obtaining realistic performance estimates in a real-world setting (Lopez-Del Rio et al., 2019; Mayr et al., 2018). Not only we accounted for it, but we also investigated if the stage in which the compound series control was introduced, in combination with the data augmentation (before or after applying SMOTE), had an impact in the outcome.

Indeed, the order had an impact in the model performance and needed careful consideration. *Resampling before clustering* solved the global imbalance of the dataset, but clustering after oversampling would lead again to a protein-wise imbalance. Analogously, *semi resampling* resampled the training and validation sets, but imbalance returned after their clustering. On the contrary, *resampling after clustering* first corrected the problem of similar compounds, and then augmented the data to reach a protein-wise balance.

Limitations and future work

This study continues our incremental work on recommendations for DL models regarding input encoding (Lopez-Del Rio et al., 2020) and control of chemical series (Lopez-Del Rio et al., 2019). While this study was limited to one architecture and four protein families, it provides a foundation to understand the basic behaviour of PCM models, insights on how to adjust performance metrics for a protein-wise analysis, and a first step towards exploring more general questions. Those could include architecture-centric analyses to confirm if the same trends are observed when changing the layers or the model structure, or using other protein families with a different distribution of actives ratios to those studied in this analysis.

6.5 CONCLUSION

Although the effect of data balance and resampling techniques had been analysed for QSAR models, it had not been studied yet in the context of proteochemometrics models, even if the bioactivity datasets used in this setting are usually imbalanced. In this paper, we have tested four different combinations of data oversampling (through SMOTE) and clustering for controlling compounds similarity. While the clustering avoids overly optimistic performance estimates, it could introduce more data imbalance (in the form of splittings having proteins with mostly active or inactive compounds). Despite this potential conflict between the resampling and the clustering, we

found that resampling was useful to improve the model behaviour and performance.

Some common performance metrics were affected by the data imbalance and yielded misleading trends. We included an imbalance-aware random baseline and defined baseline-adjusted metrics to overcome this issue, especially in F1-score and accuracy. After baseline adjustment, the metrics provided a unified picture: the largest impact in performance estimates came from the application of data augmentation to the test set (*resampling before clustering* and *resampling after clustering* outperformed *semi resampling* and *no resampling*). However, augmenting the test set may not reflect a realistic scenario.

On the other hand, *semi resampling* outperformed *no resampling* in four out of five adjusted metrics and provided a more equalized distribution of predicted actives ratio. This confirmed the data augmentation usefulness even if the data balance in the test set differed from that of the training set. This was consistent with the finding that the predicted proportion of positives of the proteochemometrics model was explained by the actual data balance in the test set, rather than that of the training set. We also found that proteins with more interactions were better predicted.

Our recommendation is thus to use the *semi resampling* strategy, i.e. clustering compounds to separate training and validation from test sets, resampling training and validation and then clustering compounds again to definitely split training and validation sets. This was carried out on the kinases protein family and further confirmed on the GPCR, NR and PR protein families. While we cannot extrapolate these results to all the proteins and imbalance distributions, this sets a sensible starting point for improving proteochemometrics modelling and remains consistent with the corresponding data imbalance studies on QSAR models.

6.6 DATA AND CODE AVAILABILITY

The bioactivity data used in our analysis is publicly available in the repository `https://github.com/Shen-Lab/DeepAffinity` (Karimi et al., 2019). The code of this analysis is publicly available at `https://github.com/b2slab/imbalance_pcm_benchmark`.

6.7 ACKNOWLEDGEMENTS

This work was supported by the Spanish Ministry of Economy and Competitiveness (www.mineco.gob.es) TEC2014-60337-R, DPI2017-89827-R, Networking Biomedical Research Centre in the subject area of Bioengineering, Biomaterials and Nanomedicine (CIBER-BBN) and Share4Rare project (Grant Agreement 780262), initiatives of Instituto de Investigación Carlos III (ISCIII). B2SLab is certified as 2017 SGR 952.

7 PUBLICATIONS AND DISCUSSION

The results were hereby summarised, by scientific publication. The last section presents a conceptual breakdown of the contributions as a whole.

7.1 IMPROVEMENT OF VALIDATION STRATEGIES

Angela Lopez-Del Rio, Alfons Nonell-Canals, David Vidal, and Alexandre Perera-Lluna (2019), "Evaluation of Cross-Validation Strategies in Sequence-Based Binding Prediction Using Deep Learning", *J. Chem. Inf. Model.* 59, 4, pp. 1645-1657, DOI: 10.1021/acs.jcim.8b00663

The effect of different cross-validation strategies on the performance of proteochemometrics models was evaluated. The architecture of the DL model was divided in a protein analysis and a compound analysis block based on state of the art models. It was trained on a benchmark dataset generated from three different publicly available sources (ChEMBL, MUV and DUD). The following four different cross-validation strategies were analysed:

- **Random** splitting
- Splitting based on k-means clustering of the compounds fingerprints (**clustering-based**)
- Splitting based on source database (**database-based**)
- Splitting based in both the clustering and the source database (**intermediate**)

These strategies were compared to a prospective validation performed with data from a later version of ChEMBL.

Results showed that the cross-validation strategy is an influential factor for model performance. The strategy with the highest compound overlap between splitting sets (random cross-validation) resulted to be over optimistic. This happens because shared scaffolds and similarity between compounds of different splitting sets inflate the performance estimates. It is a crucial finding, since random cross-validation has been the strategy most traditionally used in the literature.

Models also showed to struggle when generalizing between different databases because of data bias. This led models trained with database-based and intermediate schemes to barely outperform a random predictor (overly pessimistic estimates). Database-based was too conservative and intermediate showed that clustering on its own was not enough to render the models free of the database bias.

For its part, clustering proved to be a compromise solution, keeping predictive power without being too optimistic due to the control of data redundancy and compound similarity between splitting sets.

While amino acid sequences were one-hot encoded and padded at the end at a common length, fingerprints-based and SMILES-based representation of molecules were explored. Better performance was shown for fingerprints, although these results were inconclusive and possibly specific to the architecture of the predictive models. Additionally, the DL model was compared with a logistic regression baseline model. Surprisingly, DL models showed minimal improvement over the baseline logistic regression, even if this model is not optimal for sequential inputs.

7.2 STUDY OF THE EFFECT OF PADDING SEQUENCES

Angela Lopez-Del Rio, Maria Martin, Alexandre Perera-Lluna, and Rabie Saidi (2020), "Effect of Sequence Padding on the Performance of Protein-Based Deep Learning Models", *Sci. Rep.* 10, p. 14634, DOI: 10.1038/s41598-020-71450-8

The evaluation of validation strategies, in which the proteins were one-hot encoded, made us acknowledge the need of establishing a common length for the amino acid sequences to be the input of DL models. Although padding/truncating sequences is a required preprocessing step for most of the available DL tools nowadays, a comparison between the effect of different padding positions could not be found on the literature. Although most ML tools provide a masking feature for ignoring padded elements, it remains unavailable for some of the most common DL layers.

Thus, we performed a systematic analysis on how different padding positions affected protein-based DL models. Specifically, commonly used padding strategies were collected (pre-, post- and ext-) and tested along with four novel padding strategies (mid-, strf-, rnd-, and zoom-). Padding strategies were classified in two groups:

- **Dense paddings**: zeros are kept together (pre-, post-, ext-, mid-).
- **Sparse paddings**: zeros are interspersed on the sequence (strf-, rnd-, zoom-).

This analysis was carried on three different architectures: only feed-forward neural networks, feed-forward neural networks coupled with a convolutional layer and feed-forward neural networks coupled with a stack of convolutional layers.

The chosen task for this study was a hierarchical classification of enzymes with two levels: a binary classification of proteins into enzymes/non-enzymes, and a multi-label prediction of the enzyme type.

Our results showed that padding has indeed an effect on model performance, and convolutional layers proved to activate differently for each type of padding.

As for the specific task we analysed, there were no differences between dense types of padding, which have showed to perform better than sparse paddings for convolutional architectures. Among sparse paddings, stratified (distributing the zeros uniformly across the sequences) is the one that worked better, outperforming the rest of the padding types for the dense architecture. Finally, all the paddings were combined in an effort to achieve data augmentation, but this strategy did not improve performance.

7.3 ANALYSIS OF DATA BALANCE

Angela Lopez-del Rio, Sergio Picart-Armada, and Alexandre Perera-Lluna (2021), "Balancing Data on Deep Learning-Based Proteochemometric Activity Classification", *J. Chem. Inf. Model.* 61, 4, pp. 1657-1669, DOI: 10.1021/acs.jcim.1c00086

The evaluation of the validation strategies made us also realize the scarcity and imbalance of bioactivity data used in CADD, especially when applying DL models which require a large quantity of training data. Although data augmentation techniques are popular in many fields of application (e.g. image processing) to alleviate these problems, it is uncommon in this field. The effect of data imbalance and traditional resampling techniques had been tested for QSAR models but, up to our knowledge, an analogous study was yet lacking for proteochemometrics modeling. Additionally, existing literature did not take into account the control of the compound series bias pointed out in section 7.1.

This study provides a comprehensive analysis of the effect of data imbalance and oversampling in the performance of protein-compound activity classification DL models. Data was splitted based on compound clustering (as in section 7.1) and proteins were pre-padded according to tuning results (following recommendations in section 7.2). As opposite to section 7.2, data augmentation was used to generate synthetic compounds. Using the SMOTE oversampling technique in a per protein basis, we tested four different combinations of balancing, data clustering and splitting:

- Bioactivity data was clustered but not resampled (***no resampling***), as in section 7.1.
- ***Resampling after clustering***, in which each protein activity data in each splitting set was resampled after clustering (50% actives/50% inactives proportion).
- ***Resampling before clustering***, in which resampling was applied prior to clustering and splitting (proportion of actives/inactives within splitting set $\neq$ 50% due to the clustering).
- ***Semi resampling***, in which the test set was kept without resampling but training+validation was resampled and reclustered.

The analysis was carried on a simplified version of the two-block DL model used in section 7.1. The study was applied to the kinases protein family and GPCRs, NRs and proteases were used as external datasets, to validate the conclusions.

We showed that data augmentation was indeed useful for the target-compound activity classification task, improving the performance and alleviating the protein-wise imbalance (aggravated in the *no resampling* model predictions). We recommend the *semi resampling* strategy, given that its predictions in the test set were less imbalanced than those of *no resampling*, while keeping the original test set intact, resembling a realistic scenario.

Our study also brings light to the nature of such DL models, by showing that their protein-wise percentage of predicted actives was driven by the actual percentage of actives in the test set, rather than that of the training set.

We also found that some widespread performance metrics (especially macro-averaged F1-score and accuracy) were affected by the varying degrees of protein data imbalance. We showed how to adjust those metrics for imbalance-aware random baselines to draw sound conclusions, in line with imbalance-insensitive metrics (AUROC and MCC).

7.4 OUTCOME

The scientific output has been so far divided into three main topics based on their domain of application: data bias, proteins preprocessing and class imbalance.

- The article *Evaluation of Cross-Validation Strategies in Sequence-Based Binding Prediction Using Deep Learning* analyzes how realistic different cross-validation strategies are in DL-based proteochemometrics models. It concludes that a strategy in which compounds are clustered according to their properties and these clusters are not divided between the splitting sets achieves the most realistic performance estimates and thus, is the best way to compensate for data bias.

- The article *Effect of Sequence Padding on the Performance of Deep Learning Models in Archaeal Protein Functional Prediction* covers the complementary problem of the neglect of the padding step when using amino acid sequences as input of DL models. This paper highlights that this often overlooked step of data pre-processing deserves more consideration. Padding position should be tuned as any other hyperparameter prior to training a model.

- The article *Balancing Data on Proteochemometrics Activity Classification* investigates whether bioactivity data imbalance (combined with data clustering) could affect the DL-based protein-compound activity classification models. This paper shows that data augmentation on the training and validation sets can improve model performance and behaviour. The varying degrees of data imbalance per protein require the ad-

justment of imbalance-sensitive performance metrics by an imbalance-aware baseline.

Therefore, this thesis offers a wide perspective on the evaluation and diagnosis of DL-based target-ligand binding prediction models. Likewise, the use of explanatory models, present in every publication, offers a formal solution to the general issue of quantitative assessment of design factors.

8 CONCLUSIONS

8.1 CONCLUSION

The present doctoral thesis was motivated by the desire of improving the preprocessing and diagnosis of DL-based target-ligand binding prediction models.

- The first findings pointed to the need of acknowledging data bias when validating binding prediction models, in order to report realistic performance estimates. A cross-validation strategy based on clustering similar compounds was recommended as the best option to control for data bias.
- This work highlighted the importance of considering alternatives to the default padding option when preprocessing amino acid sequences in a DL model. Likewise, novel ways of padding the sequences were presented and characterized.
- This thesis also showed that a combination of data clustering and resampling in the training and validation test sets improved the performance of binding models. The predicted proportion of actives per protein was influenced by the actual frequencies in the test set, rather than those of the training set, suggesting that bioactivity models can alleviate the imbalance issue.

Besides, some diagnostic tools were established to interpret the results derived from DL-based binding prediction models:

- The analysis of the lower dimensional representations of a DL model through Principal Component Analysis gives insights on the underlying effect that preprocessing decisions have on the model.
- Explanatory linear models allow for quantifying the contribution of different factors to the performance estimates, such as compound encoding, model architecture, protein family, active/inactive proportion per protein, padding position and cross-validation strategy.
- Imbalance-aware random baselines avoid drawing misleading conclusions from imbalance-sensitive performance metrics, especially in bioactivity data with highly variable protein-wise active ratios. Analogously, logistic regression-like baselines should be used when quantifying the added value of novel architectures for bioactivity models.

To conclude, this thesis improved the evaluation and diagnosis of DL-based target-ligand binding prediction models. On the one hand, it provided well-founded recommendations regarding input preprocessing and

on the other hand, it introduced different tools for the correct assessment and diagnosis of these models.

8.2 FUTURE WORK

8.2.1 Cluster-based cross-validation for proteins

Cross-validation performed by clustering compound properties accounted for certain data bias issues in performance estimates. An analogous strategy may be explored for proteins, clustering them by sequence similarity, domain information or other biochemical properties. Its integration with the compound clustering should be studied.

8.2.2 Effect of padding other biological sequences

In this thesis, we have studied the effect of padding the amino acid sequence on an enzyme prediction task. This analysis might be extended to other biological sequences, such as RNA or DNA, and to other tasks. Such an effort would potentially find optimal representations for specific data domains and reveal their intrinsic biological properties.

8.2.3 Study of pharmacogenomics databases

The analyses in this thesis are framed within the context of the sequential representation of proteins in DL-based target-compound activity prediction models. In this setting, it might be explored how perturbations in the protein sequences (whether they were artificial or actual mutations) could affect the binding activity predicted by the model. Real data on activity changes for specific compounds due to a mutation would reveal whether the DL model can capture interaction terms.

8.2.4 Transformers for proteochemometrics

In the past year, many fields of application of DL, especially NLP, have embraced the use of the Transformer neural network architecture. In the context of sequential input representation, Transformers could improve the performance of CADD models. However, training these models requires larger datasets than other standard DL architectures, rendering them too demanding for our existing framework. Since public pretrained Transformers systems such as BERT have been published for NLP, we expect similar resources to eventually emerge for protein and compound spaces, which would enable their application to proteochemometrics research.

A VALIDATION STRATEGIES

EVALUATION OF CROSS-VALIDATION STRATEGIES IN SEQUENCE-BASED BINDING PREDICTION USING DEEP LEARNING

A.1 MATERIAL AND METHODS

Table S13: Detailed list of Riniker et al dataset adapted from (Riniker and G. A. Landrum, 2013). It collects 88 targets with original database, target ID, target description, number of actives and number of decoys.

Original database	Target ID	Description	Actives	Decoys
ChEMBL	100126	Serine/threonine-protein kinase B-raf	100	500
	100579	Nicotinic acid receptor 1	100	500
	10198	Voltage-gated potassium channel subunit Kv1.5	100	500
	10378	Cathepsin B	100	500
	10417	P2X purinoceptor 7	100	500
	10434	Tyrosine-protein kinase SRC	100	500
	10475	Neuropeptide Y receptor type 1	100	500
	10498	Cathepsin L	100	500
	105	Serotonin 1d (5-HT1d) receptor	100	500
	10579	C-C chemokine receptor type 4	100	500
	10752	Inhibitor of nuclear factor κ B kinase β subunit	100	500
	10773	Interleukin-8 receptor B	100	500
	10927	Urotensin II receptor	100	500
	11085	Melatonin receptor 1B	100	500
	11225	Renin	100	500
	11265	Somatostatin receptor 5	100	500
	11279	Metabotropic glutamate receptor 1	100	500
	11336	Neuropeptide Y receptor type 5	100	500
	11359	Phosphodiesterase 4D	100	500
	11442	Liver glycogen phosphorylase	100	500
	11488	Estradiol 17- β-dehydrogenase 3	100	500
	11534	Cathepsin S	100	500
	11536	Ghrelin receptor	100	500
	11575	C-C chemokine receptor type 2	100	500
	116	Oxytocin receptor	100	500
	11631	Sphingosine 1-phosphate receptor Edg-1	100	500
	11682	Glycine transporter 1	100	500
	12209	Carbonic anhydrase XII	95	500
	12252	β-secretase 1	100	500
	12261	c-Jun N-terminal kinase I	100	500
	126	Cyclooxygenase-2	100	500
	12670	Tyrosine-protein kinase receptor FLT3	100	500
	12679	C5a anaphylatoxin chemotactic receptor	100	500
	12840	Macrophage colony stimulating factor receptor	100	500
	12911	Cytochrome P450 2C9	100	500
	12968	Orexin receptor 2	100	500
	130	Dopamine receptor D3	98	500
	134	Vasopressin V1a receptor	99	500
	18061	Sodium channel protein type IX α subunit	100	500
	20014	Serine/threonine-protein kinase Aurora-A	100	500
	20174	G protein-coupled receptor 44	100	500
	219	Muscarinic acetylcholine receptor M3	99	500
	234	Insulin-like growth factor I receptor	99	500
	237	Leukotriene A4 hydrolase	100	500
	25	Glucocorticoid receptor	100	500
	276	Phosphodiesterase 4A	100	500
	28	Thymidylate synthase	100	500

This appendix reproduces the supplementary data of: Angela Lopez-del Rio, Alfons Nonell-Canals, David Vidal and Alexandre Perera-Lluna. "Evaluation of Cross-Validation Strategies in Sequence-Based Binding Prediction Using Deep Learning". *Journal of Chemical Information and Modeling, 59(4), 1645–1657 (2019).*

	36	Progesterone receptor	100	500
	43	β-2 adrenergic receptor	96	500
	8	Tyrosine-protein kinase ABL	100	500
DUD	ace	Angiotensin-converting enzyme	46	500
	ache	Acetylcholin esterase	99	500
	ar	Androgen receptor	68	500
	cdk2	Cyclin-dependent kinase 2	47	500
	cox2	Cyclooxygenase-2	212	500
	dhfr	Dihydrofolate reductase	190	500
	egfr	Epidermal growth factor receptor	365	500
	er_agonist	Estregon receptor (ER): agonists	63	500
	fgfr1	Fibroblast growth factor receptor	71	500
	fxa	Factor Xa	64	500
	gpb	Glycogen phosphorylase	49	500
	gr	Glucocorticoid receptor	32	500
	hivrt	HIV-1 RT-Rnase	34	500
	inha	Enoyl reductase	57	500
	na	Neuramidase	49	500
	p38	P38 MAP kinase	137	500
	parp	Poly(ADP-ribose) polymerase	31	500
	pdgfrb	Platelet-derived growth factor receptor β	124	500
	sahh	S-adenosylhomocysteine hydrolase	33	500
	src	Tyrosine-protein kinase C-SRC	98	500
	vegfr2	Vascular endothelial growth factor receptor 2	48	500
MUV	466	Sphingosine 1-phosphate (S_1P_1) receptor	30	500
	548	Protein kinase A (PKA)	30	500
	600	Steroidogenic factor 1 (SF_1): inhibitors	30	500
	644	Rho-kinase 2	30	500
	652	HIV-1 RT-Rnase H	30	500
	689	Ephrin receptor A4	30	500
	692	Steroidogenic factor 1 (SF_1): agonists	30	500
	712	Heat shock protein 90 (HSP90)	30	496
	713	Estregon receptor (ER) α: inhibitors	30	500
	733	Estregon receptor (ER) β	30	499
	737	Estregon receptor (ER) α: potentiators	30	499
	810	Focal adhesion kinase (FAK)	30	500
	832	Cathepsin G	30	500
	846	Factor XIa (FXIa)	30	499
	852	Factor XIIa (FXIIa)	30	498
	858	Dopamin receptor D_1	30	500
	859	Muscarinic receptor M_1	30	500

Table S14: Summary of potential applications of each cross-validation strategy.

Cross-Validation Strategy	Application
random	Make predictions on targets and compounds that are already/similar to those in the dataset.
clustering-based	Make predictions on new compounds or amino acid sequences dissimilar to those seen by the model.
database-based	Transfer learning: train the model with a database and apply it to another database.
intermediate	Transfer learning, but specifically aimed to sets of novel compounds/synthesized proteins.

A.2 RESULTS

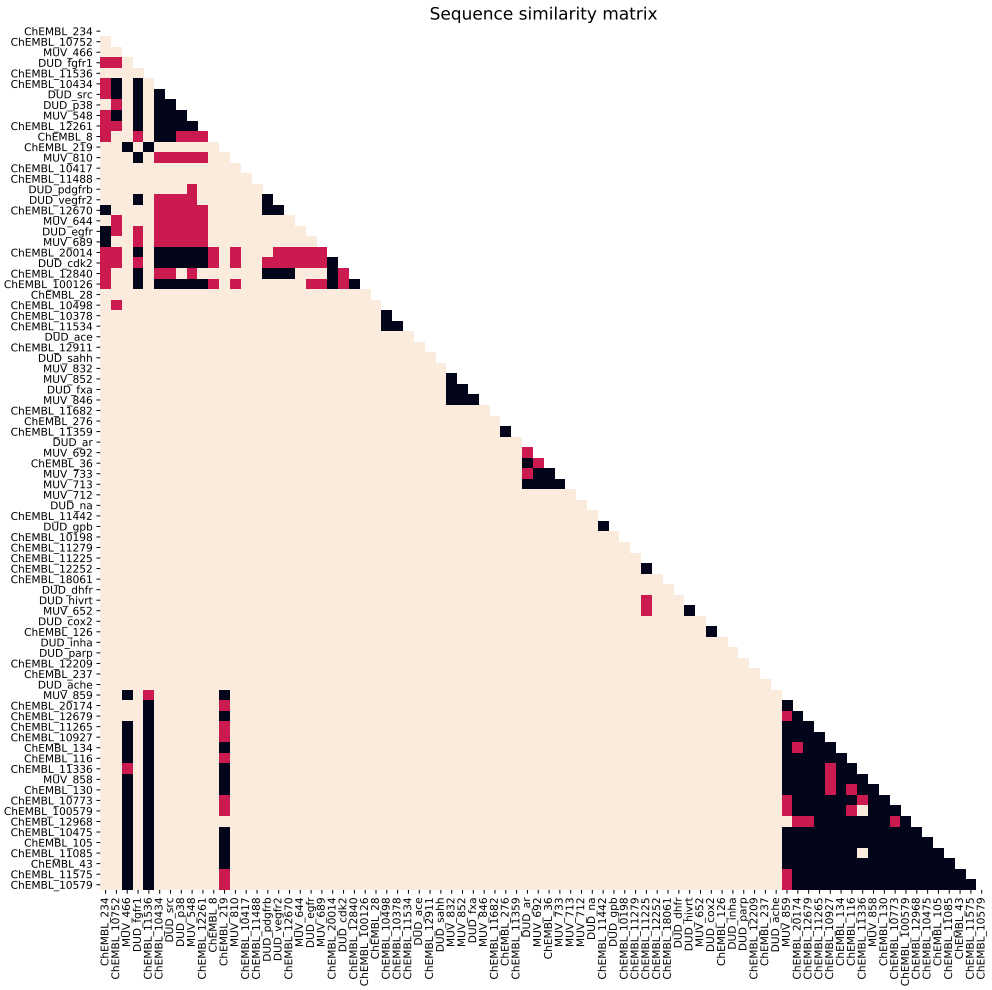

Figure S40: Sequence similarity matrix between targets of the dataset. Two targets are considered similar (dark tiles) if E-value<0.01 and there is a two-way sequence similarity of at least 30%. Pink tiles show one-way similarity. MUV_600, ChEMBL_11631, MUV_737 - ChEMBL_er_agonist and DUD_gr are not represented since they have the same sequence as MUV_692, MUV_466, MUV_713 and ChEMBL_25, respectively.

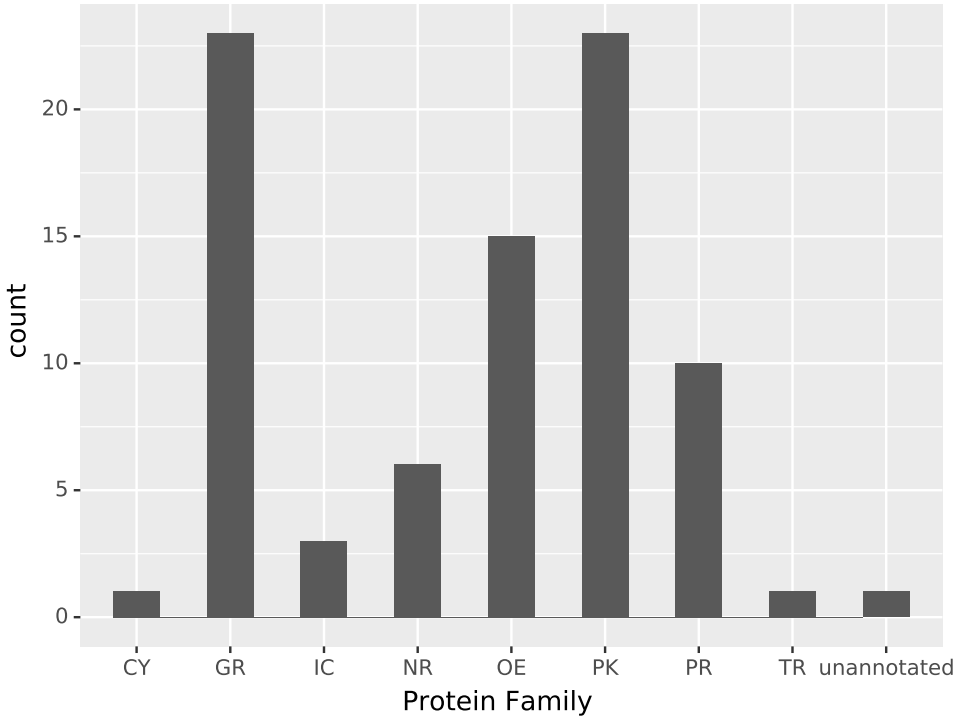

Figure S41: Histogram of protein families on the targets of the dataset.

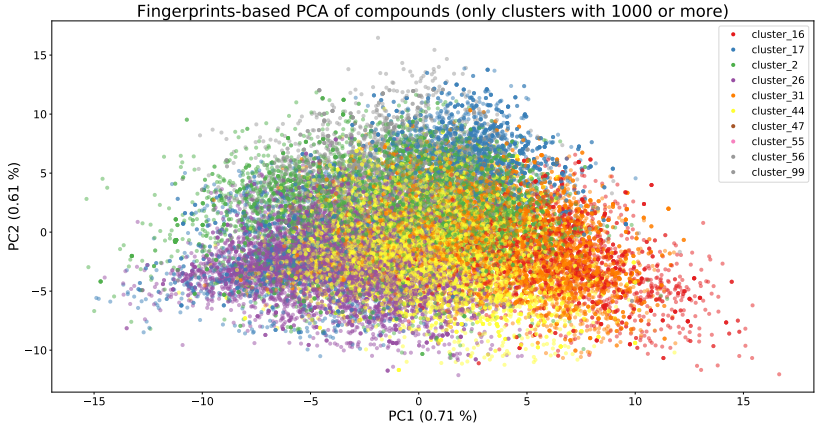

Figure S42: Fingerprints-based principal component analysis (PCA) of the most populated K-Means clusters (>1000 compounds)

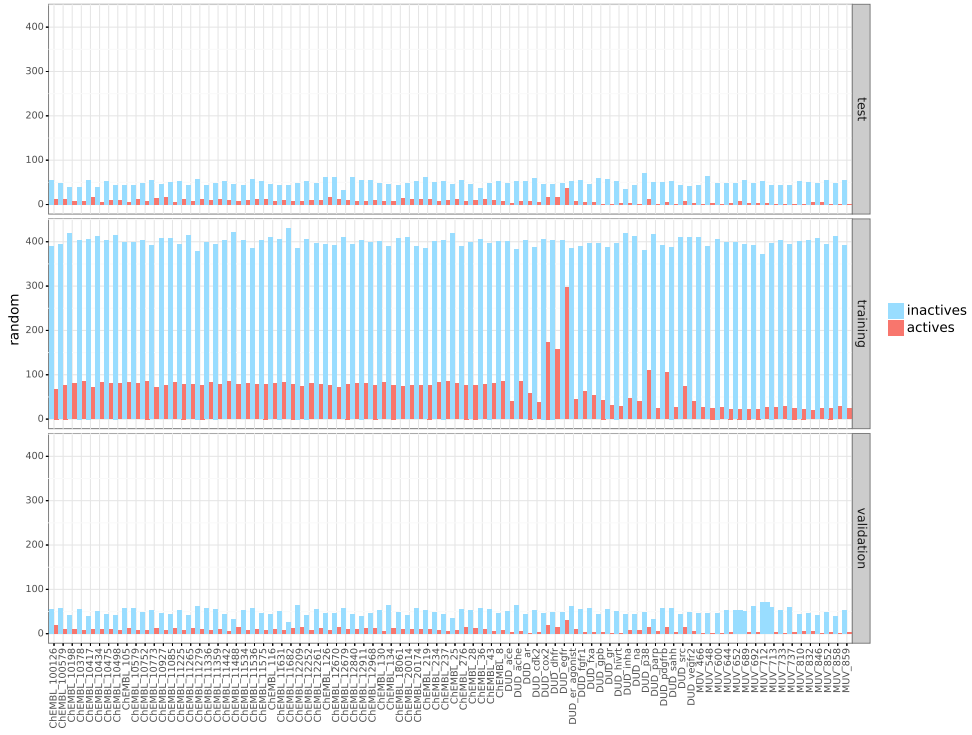

Figure S43: Distribution of active/inactive compounds for each target in training, test and validation sets in the **random** cross-validation strategy.

Figure S44: Distribution of active/inactive compounds for each target in training, test and validation sets in the **clustering** cross-validation strategy.

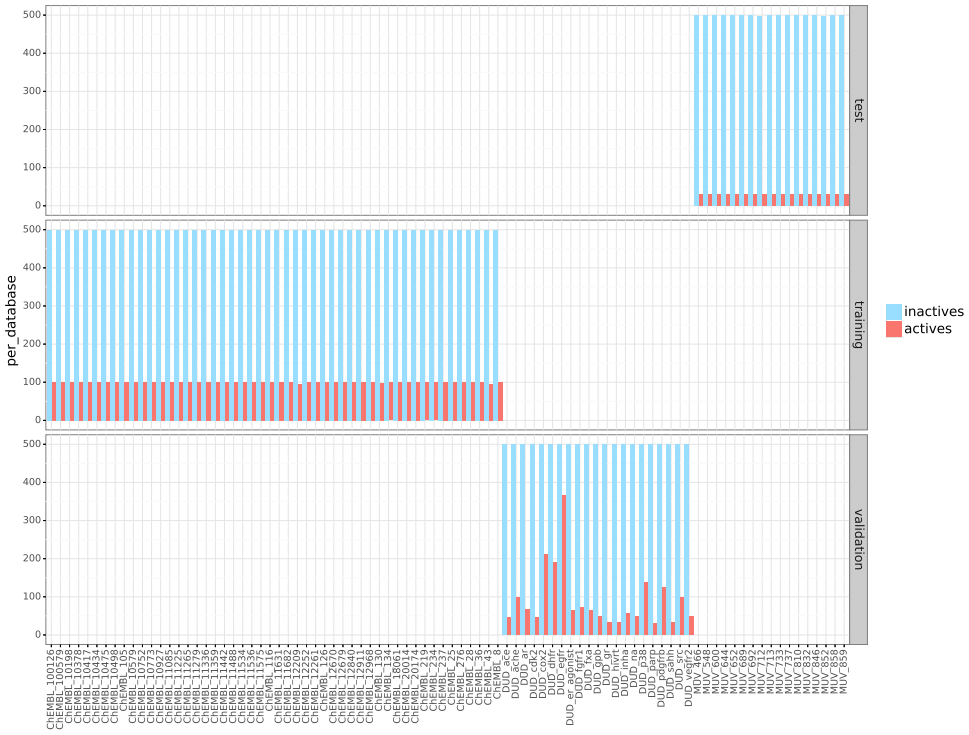

Figure S45: Distribution of active/inactive compounds for each target in training, test and validation sets in the **per_database** cross-validation strategy.

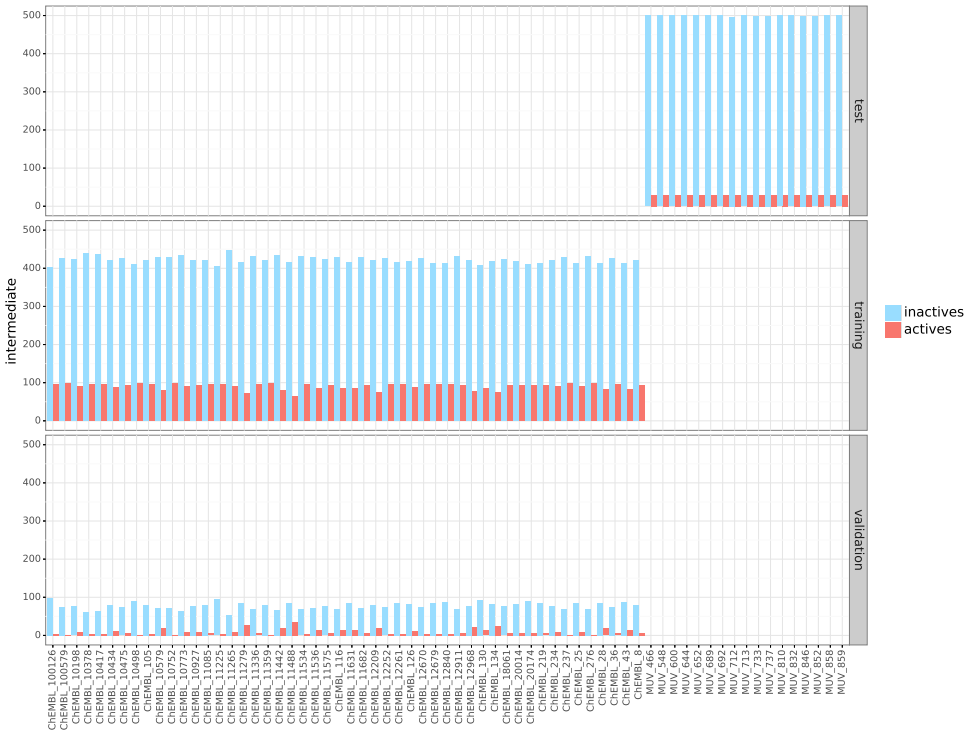

Figure S46: Distribution of active/inactive compounds for each target in training, test and validation sets in the **intermediate** cross-validation strategy. DUD targets are not represented in this distribution, since they are not present in any of the splitting groups.

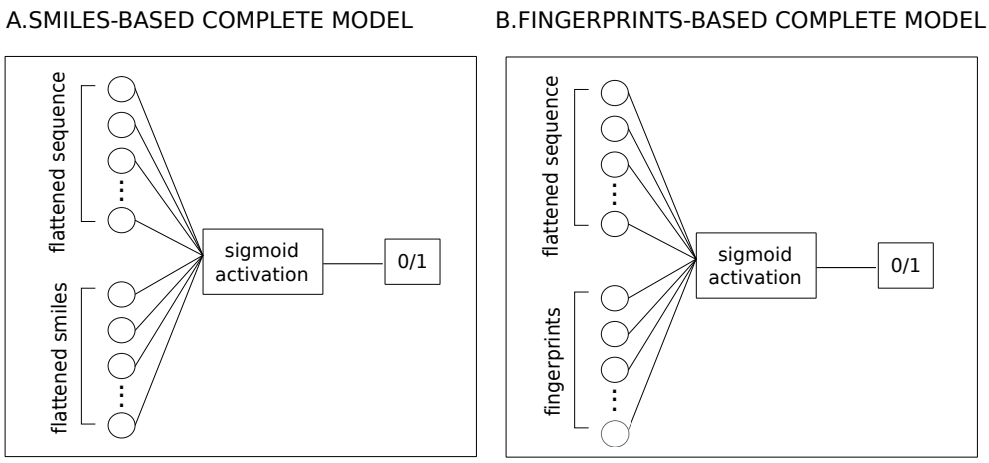

Figure S47: Schematic representation of the baseline logistic regression model **A.** When the compound is encoded by its SMILES, both amino acid sequence and SMILES inputs have to be flattened before entering the neurons of the input layer. **B.** When the compound is represented by its fingerprints, only the amino acid sequences have to be flattened.

Table S15: Description of the external common test set extracted from ChEMBL 23. It collects 22 of the 88 original targets,designated by the original ID and source database in the Riniker and Landrum dataset (See Table S13), and their number of active/inactive molecules (inactive molecules include decoys from ZINC and true inactives from ChEMBL 23).

Original database	Target ID	Actives	Inactives
ChEMBL	100126	117	71
	10378	55	194
	10498	94	275
	10773	34	70
	11085	21	42
	11279	33	216
	11534	147	294
	12209	10	20
	12252	75	160
	12670	14	29
	12911	3	74
	12968	892	2079
	20014	564	1204
	234	1	5
	8	13	30
DUD	ar	-	1
	cdk2	81	165
	dhfr	46	112
	egfr	78	156
	fgfr1	8	17
	pdgfrb	48	97
	vegfr2	256	545

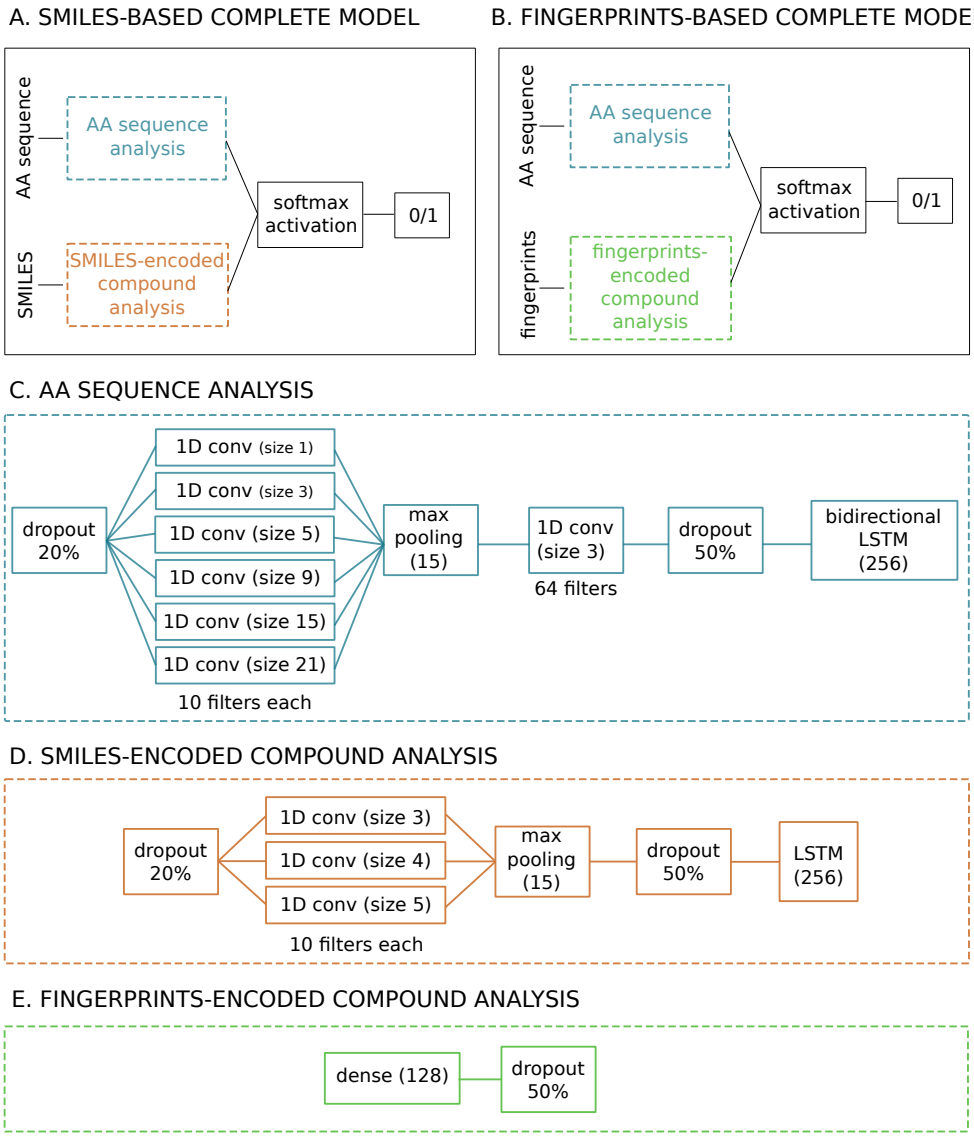

Figure S48: Schematic representation of Deep Learning models. AA: amino acids. (#) after the name of a layer refers to the number of neurons. **A.** Complete DL model when the compound is represented by their SMILES. **B.** Complete DL model when the compound is represented by their fingerprints. **C.** Amino acids sequence analysis block, a Convolutional Recurrent Neural Network architecture adapted from the model used by Jurtz et al (Jurtz et al., 2017). **D.** SMILES-encoded compound analysis block., a Convolutional Recurrent Neural Network. **E.** Fingerprints-encoded compound analysis block, a feed-forward deep neural network.

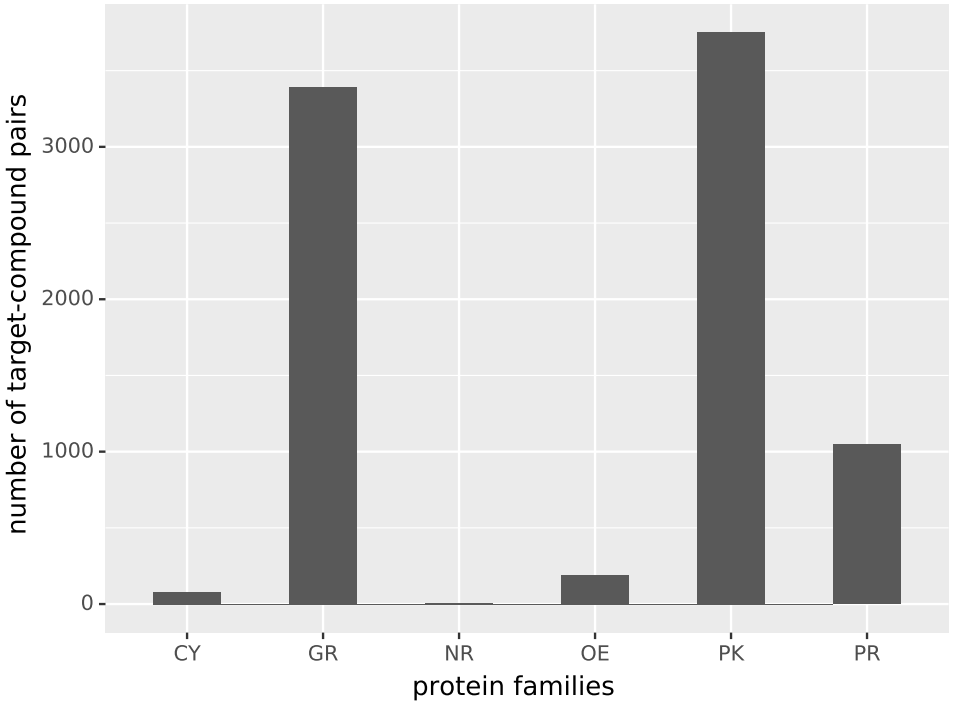

Figure S49: Distribution of protein families between the compound-target pairs on the external common test set built from ChEMBL 23.

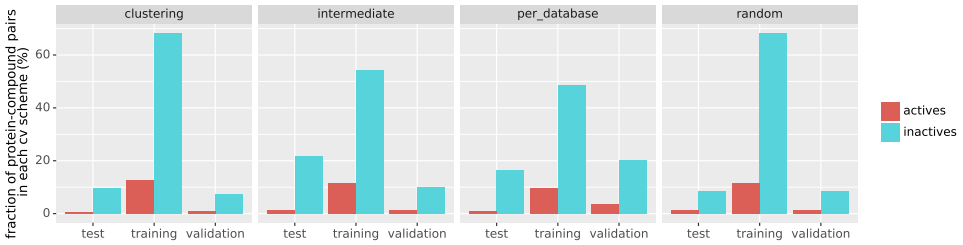

Figure S50: Percentage of active/inactive compounds in each set (training, validation and test), for each cross-validation strategy.

Figure S51: Coverage and overlap of **active** compounds between splitting sets. Numbers inside tiles refer to the percentage of overlapping compounds respect to the total number of actives, 7413.

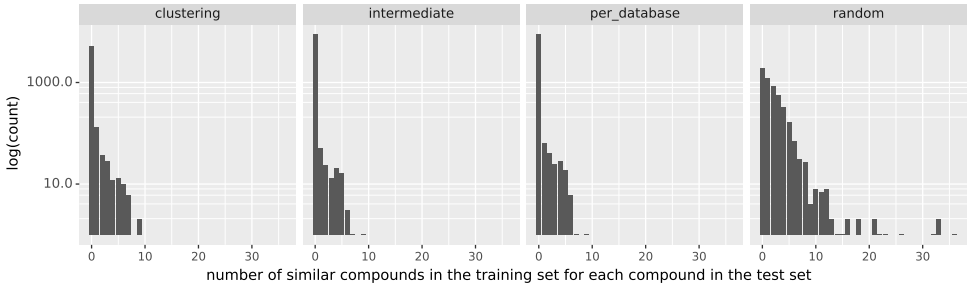

Figure S52: Histogram of number of similar compounds in the test set for each compound in the training set for each cross-validation strategy in logarithmic scale.

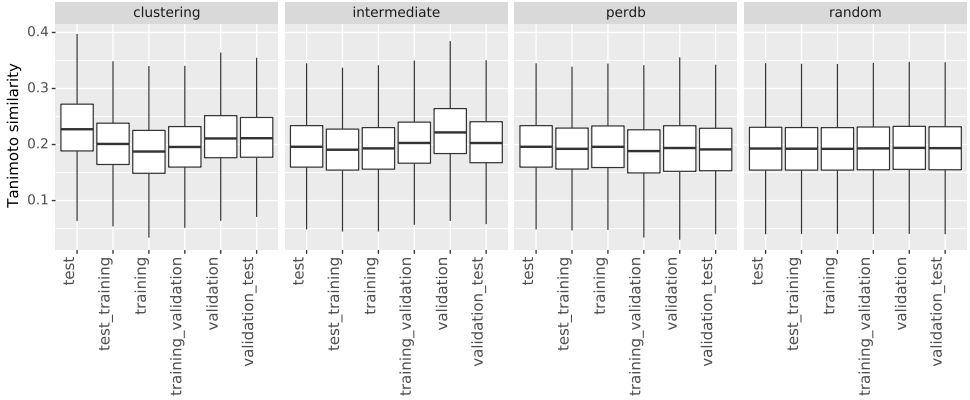

Figure S53: Tanimoto similarity between splitting sets for each cross-validation strategy. Outliers (outside the range Q1-1.5*IQR and Q3+1.5*IQR) have been excluded for a better representation.

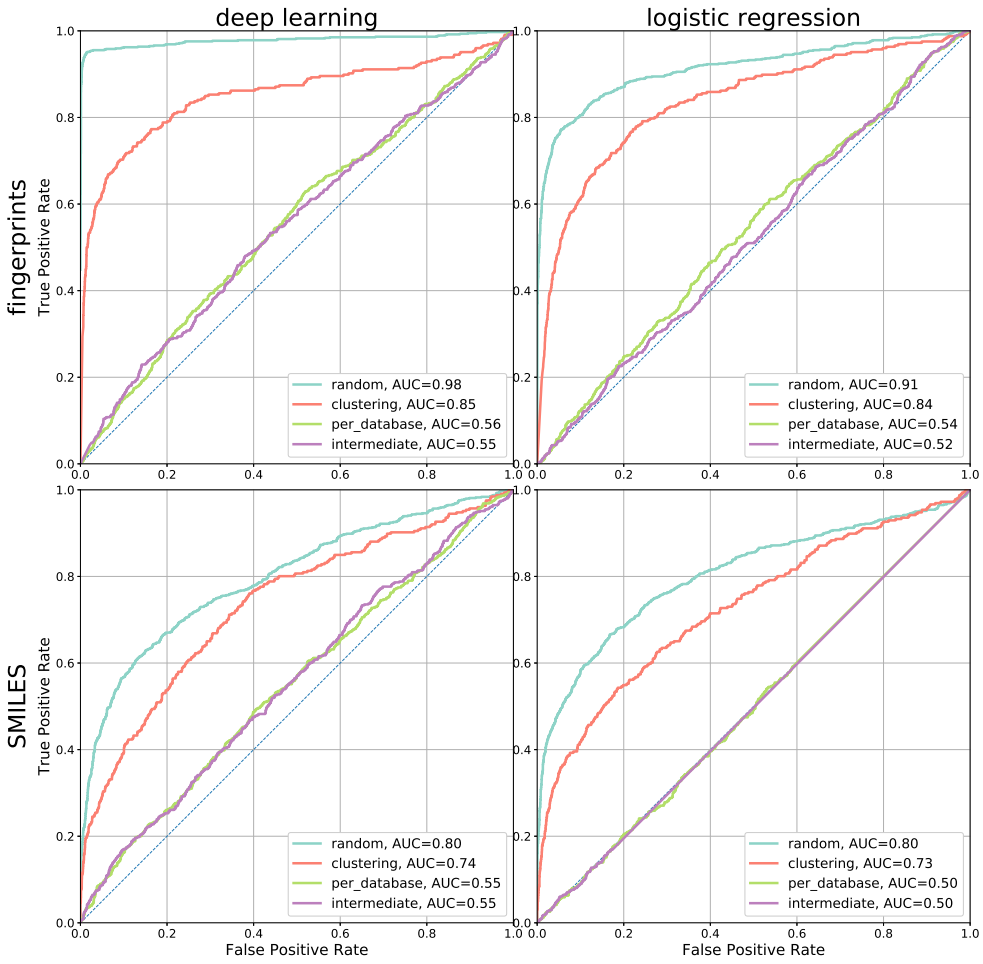

Figure S54: ROC curves of the different models and cross-validation strategies.

contrast	estimate	SE	df	t.ratio	p.value
random - clustering	0.0870	0.0236	10.0000	3.6852	0.0185
random - perdb	0.3395	0.0236	10.0000	14.3741	0.0000
random - intermediate	0.3448	0.0236	10.0000	14.5970	0.0000
clustering - perdb	0.2525	0.0236	10.0000	10.6889	0.0000
clustering - intermediate	0.2578	0.0236	10.0000	10.9118	0.0000
perdb - intermediate	0.0053	0.0236	10.0000	0.2229	0.9958

Table S16: Contrast table for the model auc ~ cv + encoding + algorithm

contrast	estimate	SE	df	t.ratio	p.value
random - clustering	0.3259	0.0466	10.0000	6.9917	0.0002
random - perdb	0.7256	0.0466	10.0000	15.5664	0.0000
random - intermediate	0.7214	0.0466	10.0000	15.4782	0.0000
clustering - perdb	0.3997	0.0466	10.0000	8.5747	0.0000
clustering - intermediate	0.3956	0.0466	10.0000	8.4865	0.0000
perdb - intermediate	-0.0041	0.0466	10.0000	-0.0883	0.9997

Table S17: Contrast table for the model bedroc20 ~ cv + encoding + algorithm

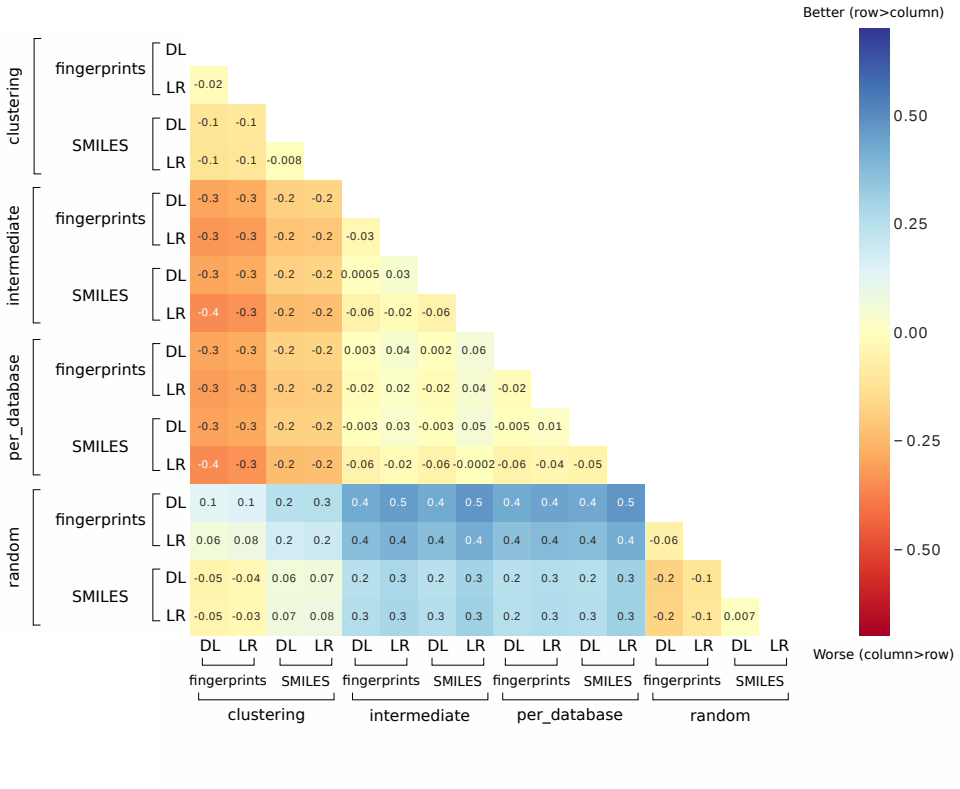

Figure S55: Mean difference for AUROC metrics between pairs of models. Crossed tiles indicate that difference for that pair of models is not statistically significant (FDR < 0.05).

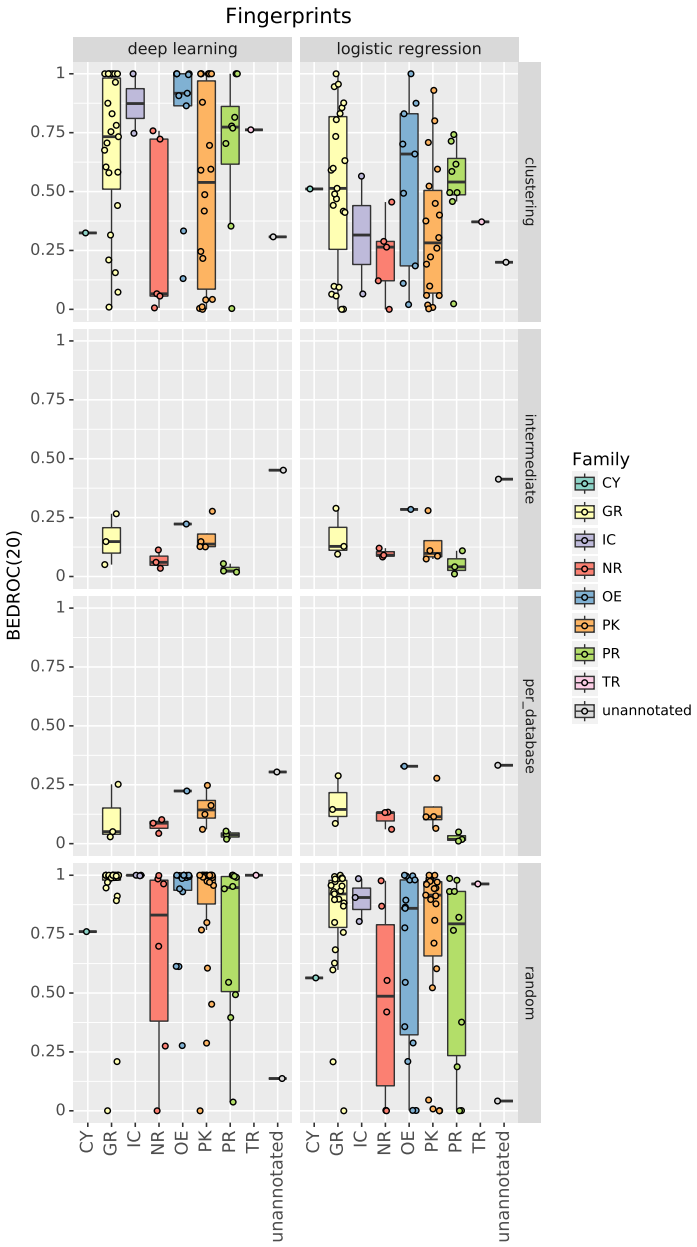

Figure S56: AUC values on test set grouped by protein families for the fingerprints models. Each data point of the figure represents a target. CY: cytochromes P450, GR: G protein-coupled receptors, IC: ion channels, NR: nuclear receptors, OE: other enzymes, PK: protein kinases, PR: proteases, TR: transporters.

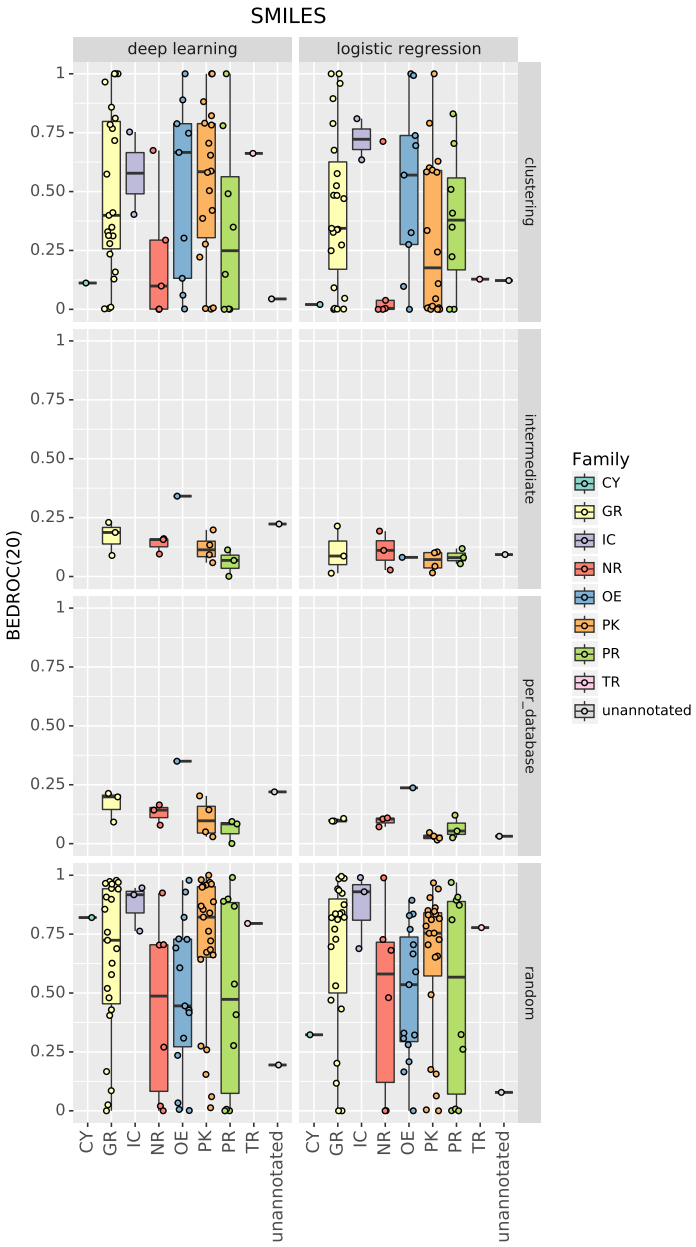

Figure S57: BEDROC(20) values on test set grouped by protein families for the SMILES models. Each data point of the figure represents a target. CY: cytochromes P450, GR: G protein-coupled receptors, IC: ion channels, NR: nuclear receptors, OE: other enzymes, PK: protein kinases, PR: proteases, TR: transporters.

Table S18: Linear regression: bedroc20 ~ cv + encoding + algorithm + family

cv clustering	−0.246*** (−0.295, −0.197)
cv perdb	−0.498*** (−0.583, −0.413)
cv intermediate	−0.485*** (−0.570, −0.400)
encoding smiles	−0.090*** (−0.134, −0.046)
algorithm dl	0.018 (−0.026, 0.062)
family GR	0.281** (0.067, 0.495)
family IC	0.354*** (0.105, 0.602)
family NR	0.069 (−0.154, 0.292)
family OE	0.238** (0.020, 0.456)
family PK	0.217** (0.003, 0.432)
family PR	0.162 (−0.057, 0.381)
family TR	0.331** (0.034, 0.628)
family unannotated	0.038 (−0.221, 0.297)
Constant	0.473*** (0.259, 0.687)
Observations	724
R^2	0.332
Adjusted R^2	0.320
Residual Std. Error	0.303 (df = 710)
F Statistic	27.172*** (df = 13; 710)
Note:	*p<0.1; **p<0.05; ***p<0.01

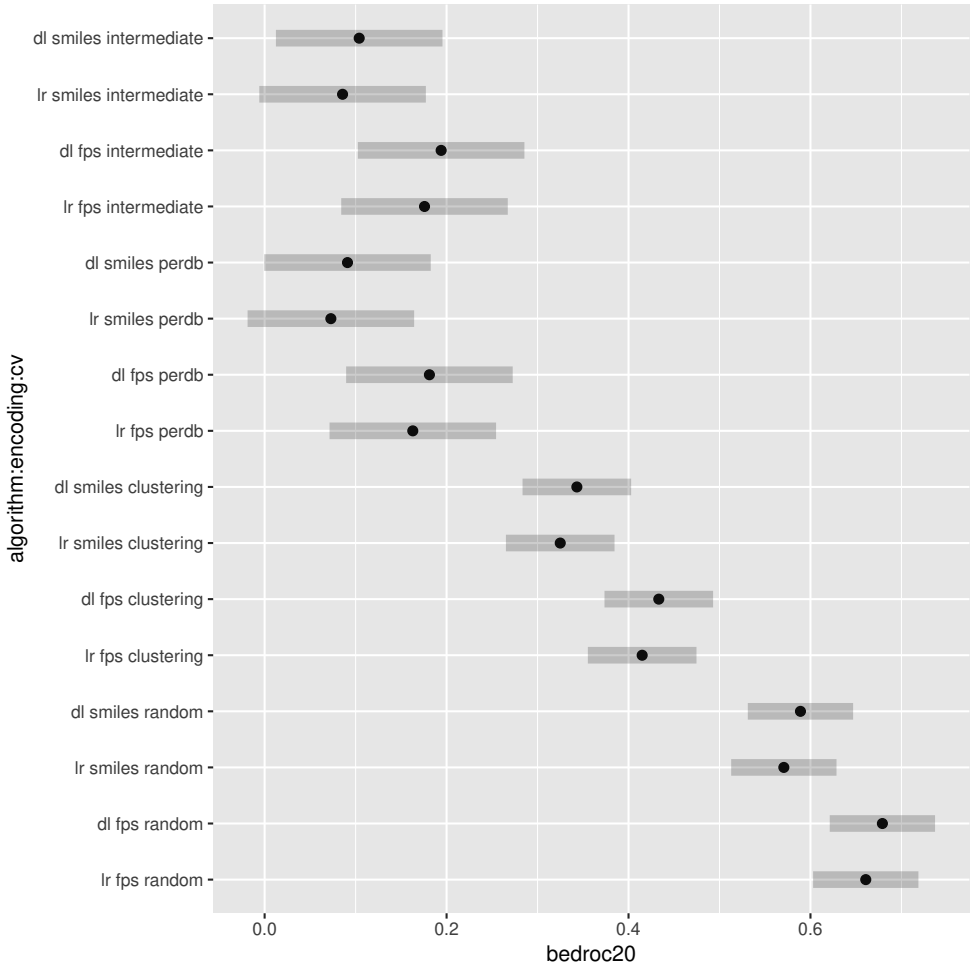

Figure S58: Confidence intervals (95%) for the predicted means using the additive model bedroc20 ~ cv + encoding + algorithm + family. The 'family' covariate was averaged over its levels.

Table S19: Linear models on the prospective AUC and BEDROC(20). The reference levels were omitted.

	AUC	BEDROC(20)
cv clustering	−0.087* (−0.175, 0.001)	−0.326*** (−0.424, −0.227)
cv perdb	−0.340*** (−0.428, −0.251)	−0.726*** (−0.824, −0.627)
cv intermediate	−0.345*** (−0.433, −0.257)	−0.721*** (−0.820, −0.623)
encoding smiles	0.001 (−0.043, 0.045)	−0.031 (−0.080, 0.019)
algorithm dl	0.023 (−0.021, 0.068)	0.050* (0.001, 0.099)
cv random:valtype prospective	−0.159*** (−0.247, −0.071)	−0.172*** (−0.270, −0.073)
cv clustering:valtype prospective	−0.065 (−0.153, 0.023)	0.162*** (0.064, 0.261)
cv perdb:valtype prospective	0.124** (0.036, 0.212)	0.548*** (0.450, 0.646)
cv intermediate:valtype prospective	0.131*** (0.043, 0.219)	0.506*** (0.407, 0.604)
Constant	0.863*** (0.793, 0.932)	0.822*** (0.744, 0.900)
Observations	32	32
R^2	0.815	0.949
Adjusted R^2	0.739	0.928
Residual Std. Error (df = 22)	0.064	0.071
F Statistic (df = 9; 22)	10.734***	45.131***

Note: *p<0.1; **p<0.05; ***p<0.01

cv	contrast	estimate	SE	df	t.ratio	p.value
random	cv - prospective	0.1590	0.0450	22.0000	3.5347	0.0019
clustering	cv - prospective	0.0653	0.0450	22.0000	1.4527	0.1604
perdb	cv - prospective	-0.1240	0.0450	22.0000	-2.7573	0.0115
intermediate	cv - prospective	-0.1307	0.0450	22.0000	-2.9055	0.0082

Table S20: Contrast table for the model auc ~ cv + encoding + algorithm + valtype:cv

cv	contrast	estimate	SE	df	t.ratio	p.value
random	cv - prospective	0.1717	0.0502	22.0000	3.4189	0.0025
clustering	cv - prospective	-0.1622	0.0502	22.0000	-3.2296	0.0039
perdb	cv - prospective	-0.5481	0.0502	22.0000	-10.9149	0.0000
intermediate	cv - prospective	-0.5055	0.0502	22.0000	-10.0674	0.0000

Table S21: Contrast table for the model bedroc20 ~ cv + encoding + algorithm + valtype:cv

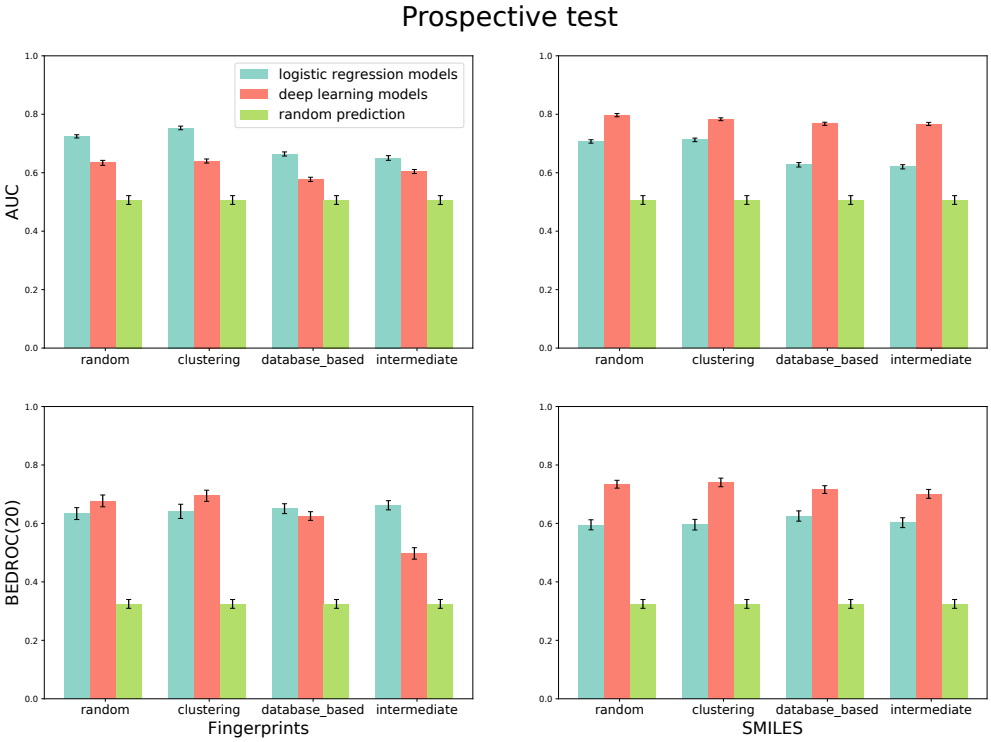

Figure S59: Comparison of performance of the different models for the **prospective test**, grouped by cross-validation strategies and colored by algorithm used for the prediction. Top row compares results in terms of AUC and bottom row in terms of BEDROC (α = 20) score. In blue, logistic regression models metrics are shown; in red, deep learning-based models, and in green, a set of random prediction synthetically generated. Left column shows results for fingerprints-encoded models and right column for SMILES-encoded models. Error bars indicate standard error of the mean from subsampled estimates.

B EFFECT OF PADDING SEQUENCES

EFFECT OF SEQUENCE PADDING ON THE PERFORMANCE OF DEEP LEARNING MODELS IN ARCHAEAL PROTEIN FUNCTIONAL PREDICTION

B.1 RESULTS

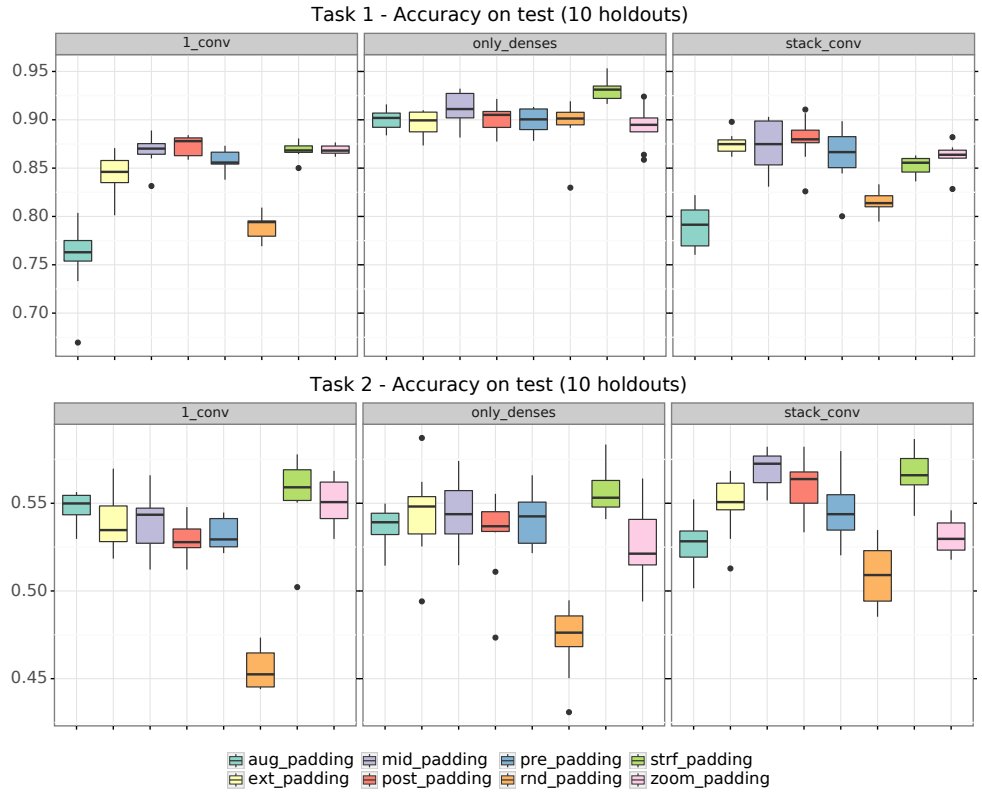

Figure S60: Accuracy on test distributions for each type of padding in each tested architecture

All linear models tables have been created with stargazer v.5.2.2 (Hlavac, 2018).

This appendix reproduces the supplementary data of: Angela Lopez-del Rio, Maria Martin, Alexandre Perera-Lluna and Rabie Saidi. "Effect of Sequence Padding on the Performance of Deep Learning Models in Archaeal Protein Functional Prediction". *Scientific Reports, 10, 14634 (2020).*

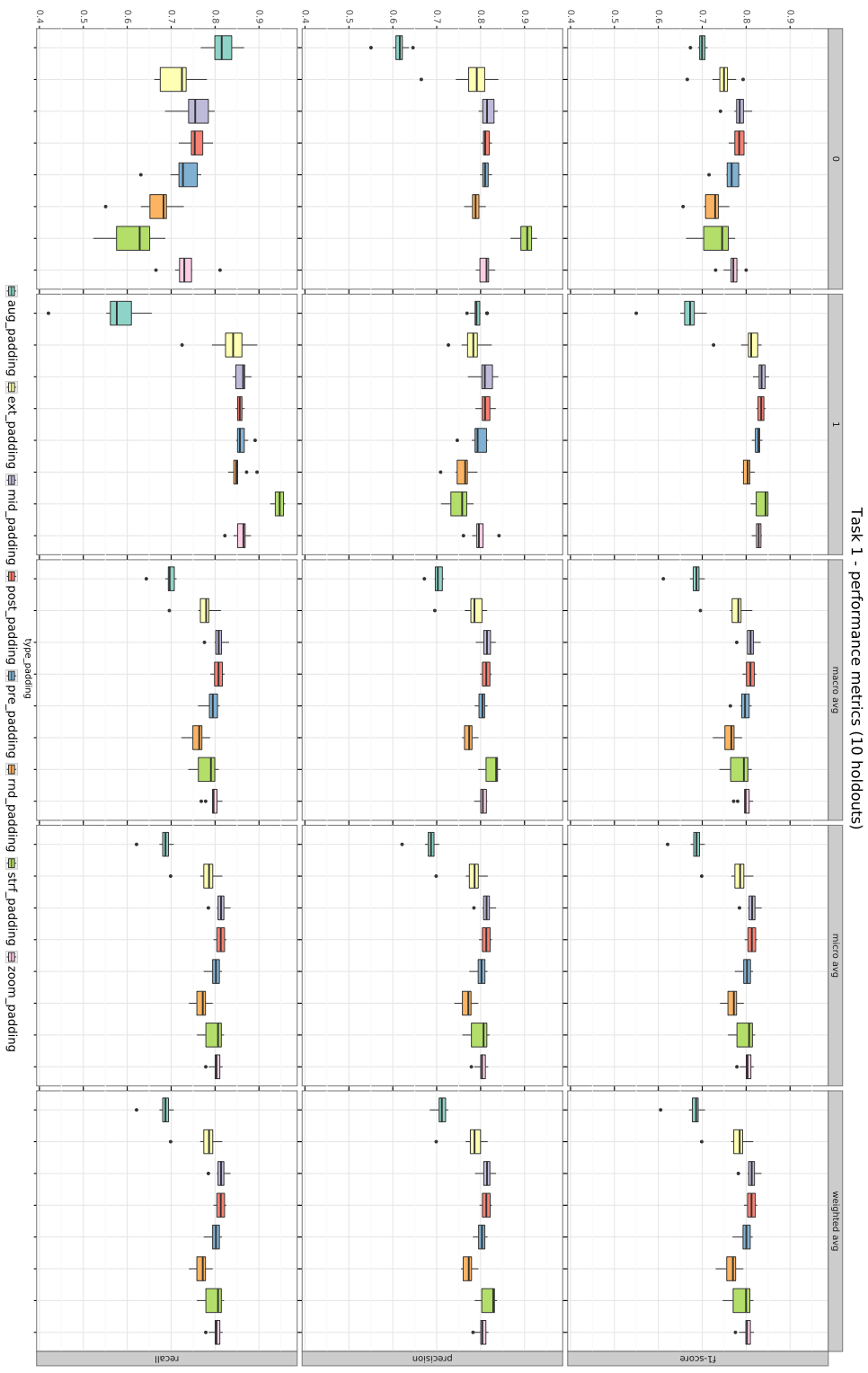

Figure S61: Task 1 F1-score measurements detailed for each label (0: non-enzyme, 1: enzyme) **only_denses** architecture.

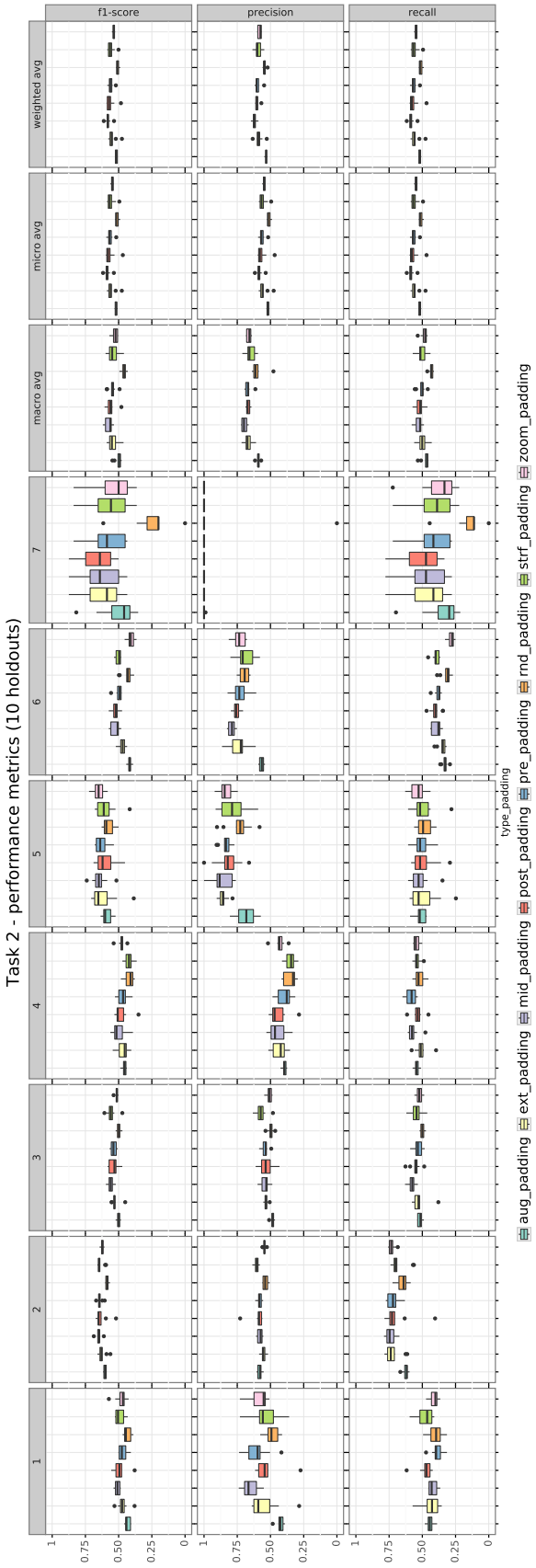

Figure S62: Task 2 F1-score measurements detailed for each label (1-7: enzyme classes) in **only_denses** architecture.

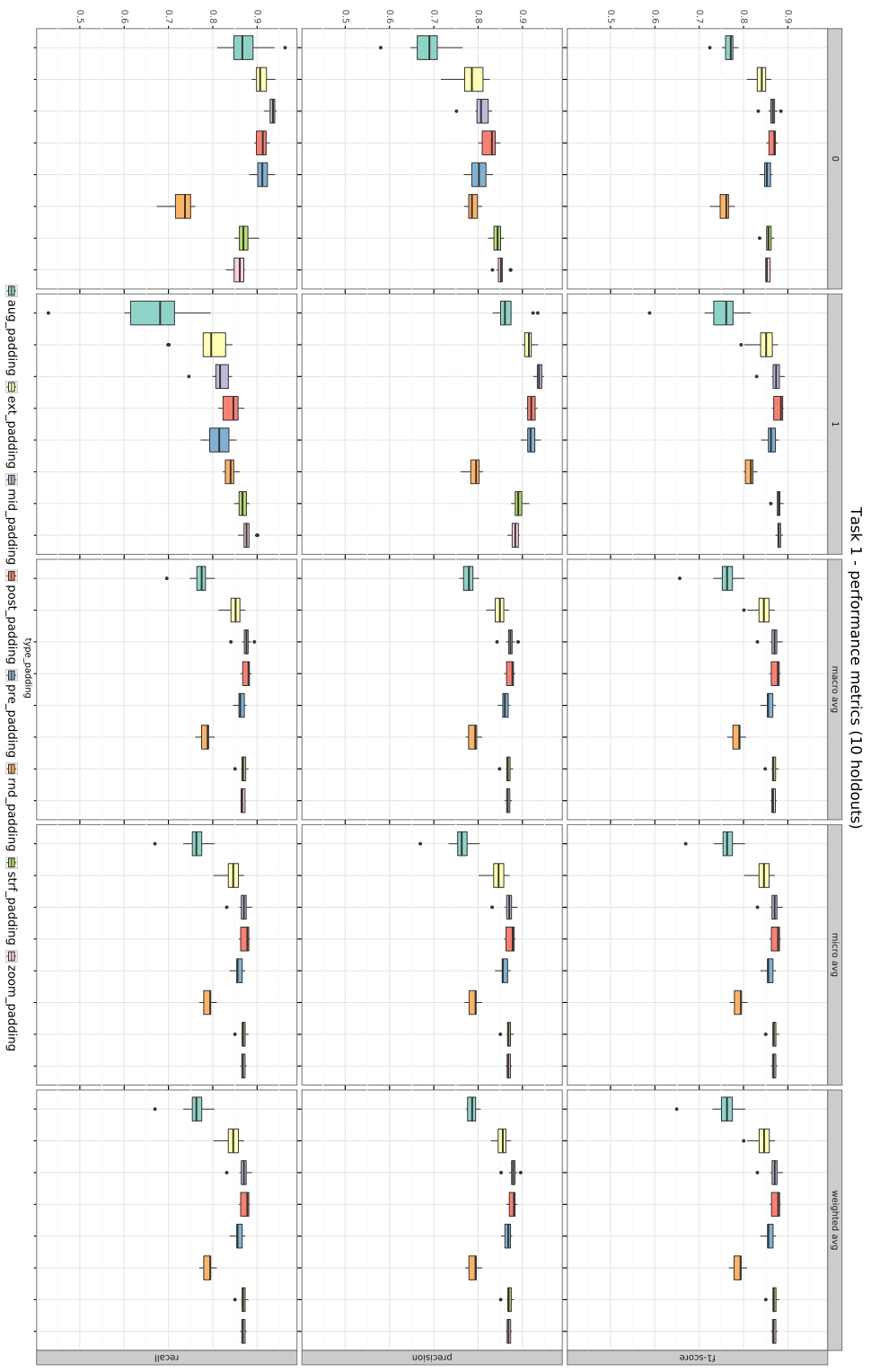

Figure S63: Task 1 F1-score measurements detailed for each label (0: non-enzyme, 1: enzyme) in **1_conv** architecture.

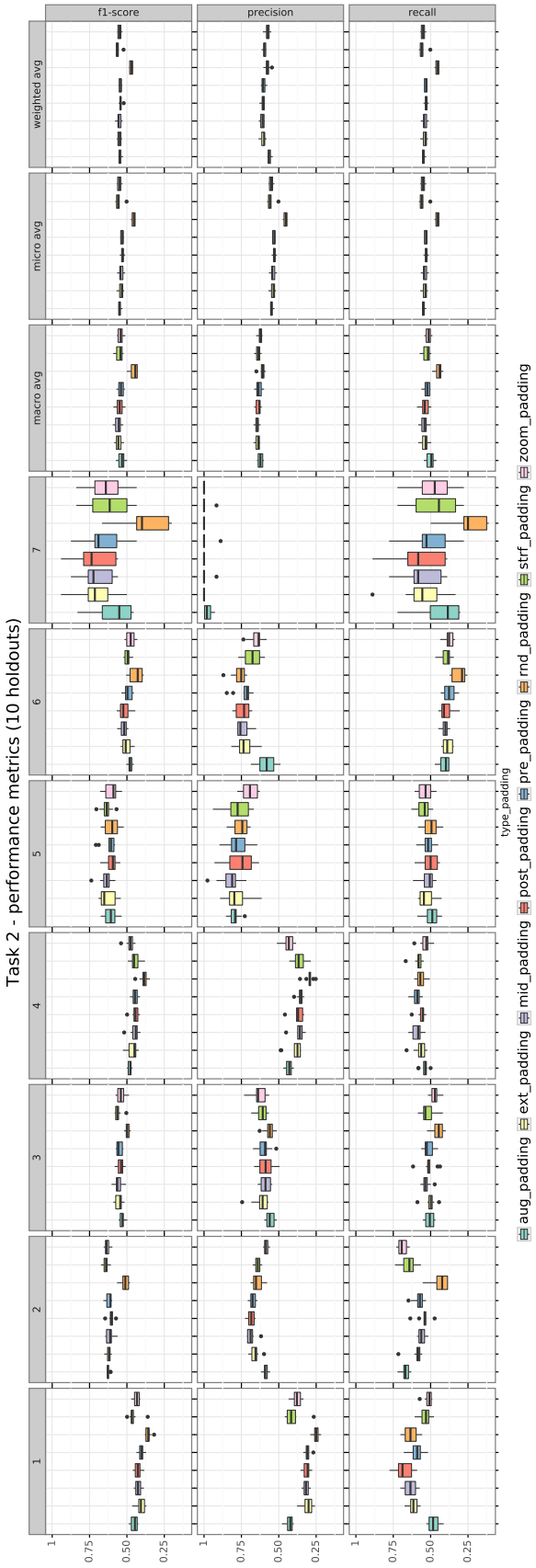

Figure S64: Task 2 F1-score measurements detailed for each label (1-7: enzyme classes) in **1_conv** architecture.

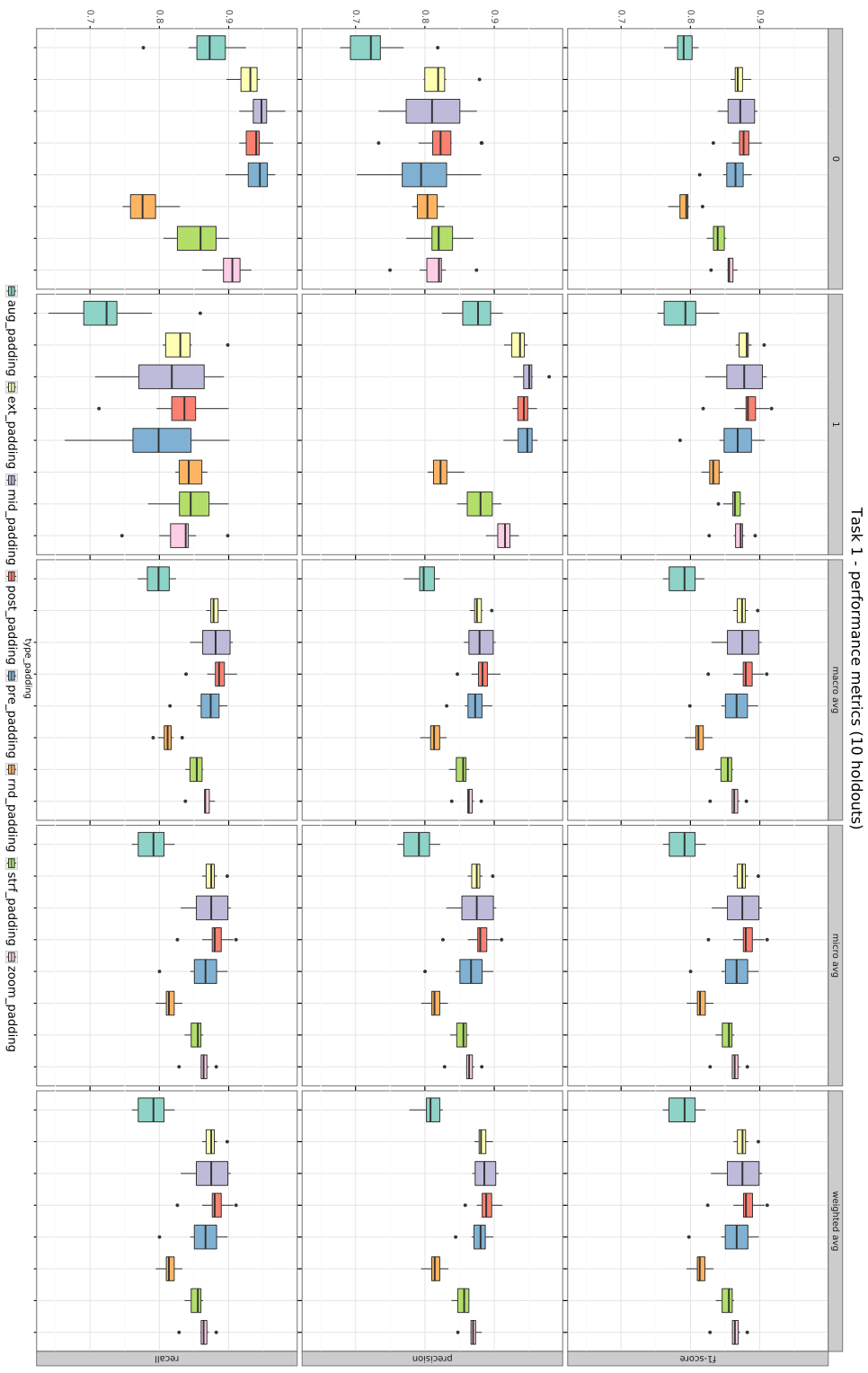

Figure S65: Task 1 F1-score measurements detailed for each label (0: non-enzyme, 1: enzyme) in **stack_conv** architecture.

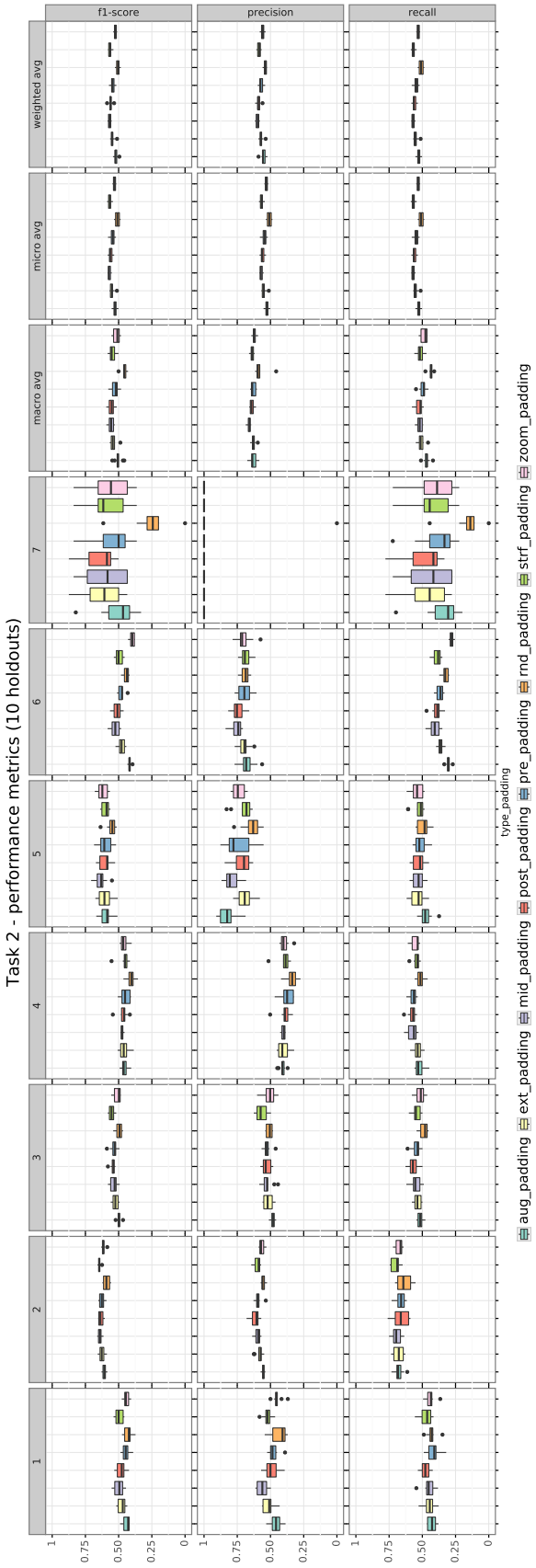

Figure S66: Task 2 F1-score measurements detailed for each label (1-7: enzyme classes) in **stack_conv** architecture.

Table S22: Full additive linear models on F1-score. The reference levels were omitted.

	Task 1	Task 2
architecture1_conv	−0.063*** (−0.069, −0.057)	0.021*** (0.011, 0.030)
architecturestack_conv	−0.052*** (−0.058, −0.046)	0.010** (0.0004, 0.020)
enz_type1	0.012*** (0.008, 0.017)	
enz_type2		0.167*** (0.152, 0.182)
enz_type3		0.097*** (0.082, 0.111)
enz_type4		0.008 (−0.007, 0.023)
enz_type5		0.167*** (0.152, 0.182)
enz_type6		0.044*** (0.029, 0.059)
enz_type7		0.100*** (0.085, 0.115)
type_paddingaug_padding	−0.069*** (−0.078, −0.059)	−0.014* (−0.029, 0.002)
type_paddingext_padding	−0.013*** (−0.023, −0.003)	0.008 (−0.008, 0.024)
type_paddingmid_padding	0.0001 (−0.010, 0.010)	0.015* (−0.001, 0.031)
type_paddingpre_padding	−0.011** (−0.020, −0.001)	−0.001 (−0.017, 0.015)
type_paddingrnd_padding	−0.053*** (−0.062, −0.043)	−0.085*** (−0.101, −0.069)
type_paddingstrf_padding	−0.001 (−0.011, 0.009)	0.011 (−0.005, 0.027)
type_paddingzoom_padding	−0.010* (−0.019, −0.00001)	−0.033*** (−0.049, −0.017)
Intercept	0.916*** (0.908, 0.924)	0.444*** (0.428, 0.460)
Observations	480	1,680
R^2	0.666	0.427
Adjusted R^2	0.659	0.422
Residual Std. Error	0.027 (df = 469)	0.083 (df = 1664)
F Statistic	93.390*** (df = 10; 469)	82.834*** (df = 15; 1664)
Note:		*p<0.1; **p<0.05; ***p<0.01

Table S23: Linear model on F1-score to analyze if padding position affects performance. The reference levels were omitted.

	Task 1	Task 2
enz_type1	0.006 (−0.005, 0.016)	
enz_type2		0.157*** (0.130, 0.184)
enz_type3		0.060*** (0.033, 0.087)
enz_type4		−0.013 (−0.040, 0.014)
enz_type5		0.136*** (0.109, 0.163)
enz_type6		0.024* (−0.003, 0.051)
enz_type7		0.125*** (0.098, 0.152)
type_paddingext_padding	−0.004 (−0.018, 0.010)	−0.014 (−0.034, 0.007)
type_paddingmid_padding	−0.006 (−0.020, 0.009)	0.004 (−0.017, 0.024)
type_paddingpre_padding	−0.017** (−0.031, −0.002)	−0.027** (−0.047, −0.006)
Intercept	0.876*** (0.865, 0.888)	0.484*** (0.461, 0.507)
Observations	80	280
R^2	0.083	0.538
Adjusted R^2	0.034	0.523
Residual Std. Error	0.023 (df = 75)	0.062 (df = 270)
F Statistic	1.697 (df = 4; 75)	34.933*** (df = 9; 270)
Note:		*p<0.1; **p<0.05; ***p<0.01

Table S24: Linear model on F1-score to analyze what is the effect of switching between dense paddings. The reference levels were omitted.

	Task 1	Task 2
enz_type1	0.006 (−0.005, 0.016)	
enz_type2		0.157*** (0.130, 0.184)
enz_type3		0.060*** (0.033, 0.087)
enz_type4		−0.013 (−0.040, 0.014)
enz_type5		0.136*** (0.109, 0.163)
enz_type6		0.024* (−0.003, 0.051)
enz_type7		0.125*** (0.098, 0.152)
type_paddingext_padding	−0.004 (−0.018, 0.010)	−0.014 (−0.034, 0.007)
type_paddingmid_padding	−0.006 (−0.020, 0.009)	0.004 (−0.017, 0.024)
type_paddingpre_padding	−0.017** (−0.031, −0.002)	−0.027** (−0.047, −0.006)
Intercept	0.876*** (0.865, 0.888)	0.484*** (0.461, 0.507)
Observations	80	280
R^2	0.083	0.538
Adjusted R^2	0.034	0.523
Residual Std. Error	0.023 (df = 75)	0.062 (df = 270)
F Statistic	1.697 (df = 4; 75)	34.933*** (df = 9; 270)
Note:		*p<0.1; **p<0.05; ***p<0.01

Table S25: Linear model on F1-score to analyze what is the effect of changing from the standard dense padding to sparse paddings. The reference levels were omitted.

	Task 1	Task 2
enz_type1	0.022*** (0.014, 0.029)	
enz_type2		0.158*** (0.125, 0.192)
enz_type3		0.064*** (0.030, 0.097)
enz_type4		−0.015 (−0.048, 0.018)
enz_type5		0.132*** (0.099, 0.165)
enz_type6		−0.001 (−0.034, 0.032)
enz_type7		0.050*** (0.016, 0.083)
type_paddingrnd_padding	−0.067*** (−0.078, −0.056)	−0.098*** (−0.123, −0.073)
type_paddingstrf_padding	−0.028*** (−0.039, −0.017)	−0.007 (−0.032, 0.018)
type_paddingzoom_padding	−0.017*** (−0.028, −0.006)	−0.038*** (−0.063, −0.013)
Intercept	0.868*** (0.860, 0.877)	0.498*** (0.470, 0.526)
Observations	80	280
R^2	0.720	0.497
Adjusted R^2	0.705	0.481
Residual Std. Error	0.017 (df = 75)	0.076 (df = 270)
F Statistic	48.257*** (df = 4; 75)	29.692*** (df = 9; 270)
Note:		*p<0.1; **p<0.05; ***p<0.01

Table S26: Linear model on F1-score to analyze if an ensemble of paddings is beneficial. The reference levels were omitted.

	Task 1	Task 2
architecture1_conv	−0.034*** (−0.048, −0.020)	0.032** (0.007, 0.057)
enz_type2		0.166*** (0.144, 0.188)
enz_type3		0.093*** (0.071, 0.115)
enz_type4		0.003 (−0.019, 0.025)
enz_type5		0.154*** (0.132, 0.176)
enz_type6		0.039*** (0.016, 0.061)
enz_type7		0.124*** (0.102, 0.146)
architectureonly_denses	0.110*** (0.096, 0.123)	0.027** (0.002, 0.052)
enz_type1	0.009*** (0.003, 0.015)	
type_paddingpost_padding	0.089*** (0.076, 0.103)	0.050*** (0.025, 0.075)
type_paddingstrf_padding	0.061*** (0.048, 0.075)	0.043*** (0.018, 0.068)
architecture1_conv:type_paddingpost_padding	0.028*** (0.008, 0.047)	−0.036** (−0.071, −0.0004)
architectureonly_denses:type_paddingpost_padding	−0.089*** (−0.108, −0.070)	−0.072*** (−0.108, −0.037)
architecture1_conv:type_paddingstrf_padding	0.050*** (0.031, 0.069)	−0.029 (−0.064, 0.007)
architectureonly_denses:type_paddingstrf_padding	−0.031*** (−0.050, −0.012)	−0.026 (−0.061, 0.010)
Intercept	0.786*** (0.775, 0.796)	0.421*** (0.399, 0.444)
Observations	180	630
R^2	0.859	0.444
Adjusted R^2	0.851	0.431
Residual Std. Error	0.022 (df = 170)	0.076 (df = 615)
F Statistic	114.720*** (df = 9; 170)	35.043*** (df = 14; 615)
Note:		*p<0.1; **p<0.05; ***p<0.01

Table S27: Linear model on F1-score to analyze how might enzyme type affect differently the performances of some padding types. The reference levels were omitted.

	Task 1	Task 2
architecture1_conv	0.005 (−0.003, 0.013)	−0.0002 (−0.019, 0.019)
enz_type2		0.162*** (0.120, 0.204)
enz_type3		0.098*** (0.056, 0.140)
enz_type4		−0.003 (−0.044, 0.039)
enz_type5		0.141*** (0.100, 0.183)
enz_type6		0.075*** (0.033, 0.116)
enz_type7		0.140*** (0.099, 0.182)
architectureonly_denses	0.050*** (0.042, 0.058)	−0.022** (−0.041, −0.002)
enz_type1	0.009* (−0.001, 0.018)	
type_paddingstrf_padding	−0.006 (−0.015, 0.003)	0.030 (−0.012, 0.072)
enz_type1:type_paddingstrf_padding	0.010 (−0.004, 0.023)	
enz_type2:type_paddingstrf_padding		−0.006 (−0.065, 0.053)
enz_type3:type_paddingstrf_padding		−0.009 (−0.068, 0.050)
enz_type4:type_paddingstrf_padding		−0.036 (−0.095, 0.023)
enz_type5:type_paddingstrf_padding		0.001 (−0.058, 0.059)
enz_type6:type_paddingstrf_padding		−0.047 (−0.106, 0.011)
enz_type7:type_paddingstrf_padding		−0.034 (−0.092, 0.025)
Intercept	0.861*** (0.853, 0.869)	0.457*** (0.426, 0.489)
Observations	120	420
R^2	0.629	0.405
Adjusted R^2	0.613	0.383
Residual Std. Error	0.019 (df = 114)	0.082 (df = 404)
F Statistic	38.674*** (df = 5; 114)	18.319*** (df = 15; 404)
Note:		*p<0.1; **p<0.05; ***p<0.01

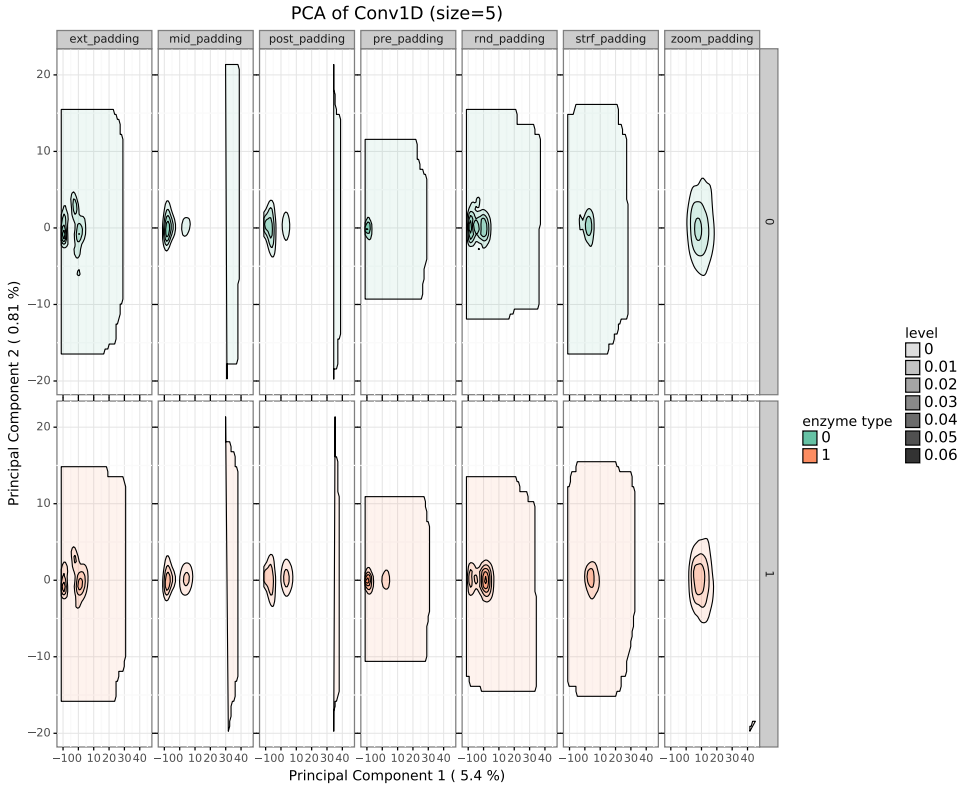

Figure S67: Density representation of PC1 vs PC2 of the activations of the 1D Convolutional layer of1conv model for each type of padding in task 1.

In Figure S67, activations seem to be more spread along the PC space than in Figure 29, the analogous representation for task 2. This happens for all types of padding except for zoom-, whose distribution is very similar to that of zoom-padding for task 2.

In order to quantify the differences observed in the first principal component of the activations of the convolutional filters for task 2 models, we built the explanatory model explained by Equation S18. Results can be seen in Table S28. All terms are statistically significant, which means there are differences for PC1 according to enzyme type and padding type. Terms for enzyme types 2, 3 and 4 are negative while they are positive for 1, 5, 6 and 7.

$$\text{PC1} \sim \text{enzyme\_type} + \text{type\_padding} \tag{S18}$$

Table S28: Linear model on PC1 of the activations. The reference levels were omitted.

paddingext_padding	−0.007 (−0.271, 0.258)
paddingmid_padding	−0.524*** (−0.789, −0.259)
paddingpre_padding	−1.181*** (−1.445, −0.916)
paddingrnd_padding	6.355*** (6.091, 6.620)
paddingstrf_padding	7.893*** (7.628, 8.158)
paddingzoom_padding	6.473*** (6.209, 6.738)
enzyme_type2	−2.425*** (−2.690, −2.161)
enzyme_type3	−1.305*** (−1.570, −1.041)
enzyme_type4	−1.835*** (−2.100, −1.570)
enzyme_type5	0.960*** (0.695, 1.224)
enzyme_type6	3.128*** (2.864, 3.393)
enzyme_type7	1.582*** (1.318, 1.847)
Intercept	−2.731*** (−2.986, −2.476)
Observations	62,720
R^2	0.172
Adjusted R^2	0.172
Residual Std. Error	9.041 (df = 62707)
F Statistic	1,085.138*** (df = 12; 62707)
Note:	*p<0.1; **p<0.05; ***p<0.01

B.2 DATA DISTRIBUTION

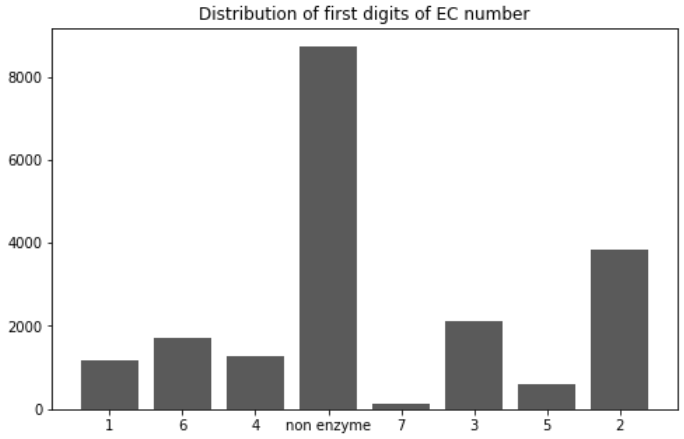

Figure S68: Distribution of enzyme types in the Archaea UniprotKB proteins.

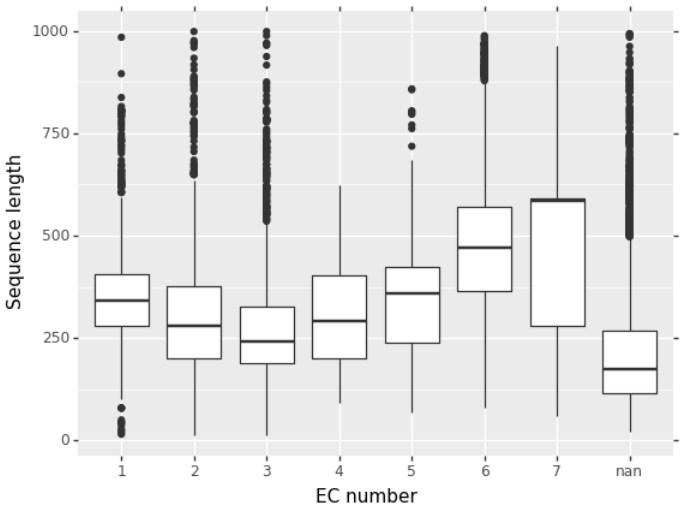

Figure S69: Sequence length stratified by enzyme type (according to the first digit of the EC number). 'nan' refers to proteins without EC number (non-enzymes). $p - value = 0.0$ for Kruskal-Wallis H-test for independent samples, so sequence length differences between different enzyme types are significant.

B.3 MODELS ARCHITECTURE

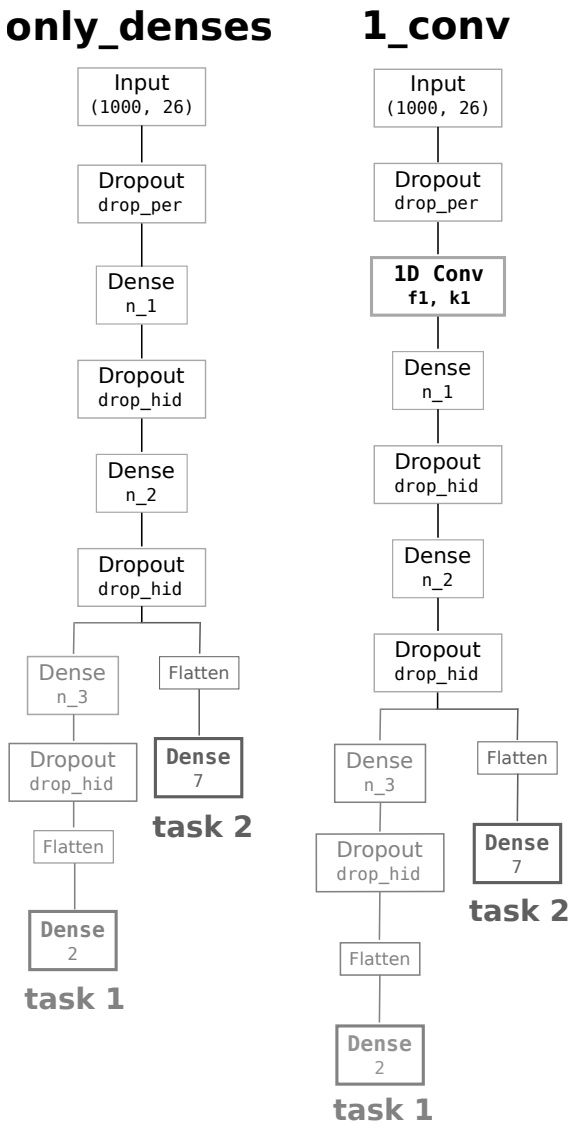

Figure S70: Schematic representation of two of the tested architectures: **only_denses** and **1_conv**. Each one of the architectures comprises two different models: task 1 and task 2.

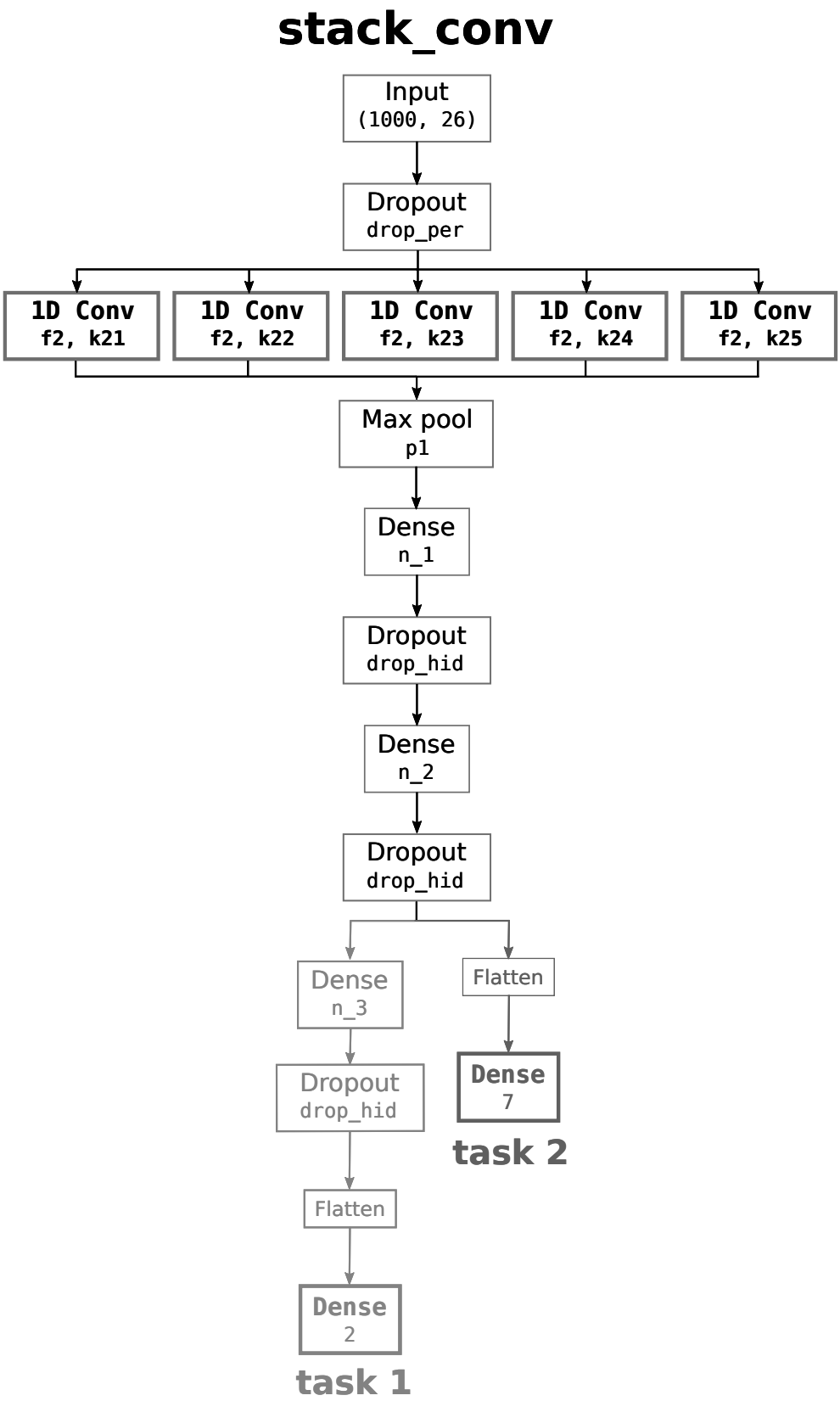

Figure S71: Schematic representation of **stack_conv** architecture. Each one of the architectures comprises two different models: task 1 and task 2.

The set of values for the number of neurons in the feed-forward part of the model, which is common for the four architectures, is based on the number

of enzyme classes and subclasses. There are 7 classes of enzymes according to the first digit of the EC number (8 if we take count non-enzymes), 76 subclasses according to the second digit of the EC number (77 with non-enzymes) and 313 categories according to the third digit of the EC number (314 with the non-enzymes). So the values for the feed-forward layers would be 314, 77 and 8, respectively. These values aim to have a biological meaning since deep neural networks are able to extract hierarchical feature representations and enzyme classification has a tree-structured label space.

We tried to fit a bi-LSTM model to also test the effect of padding on this architecture. However, this model was too complex to converge within the range of parameters of the other three architectures (number of epochs, optimizer, learning rate). As stated by Li et al, 2018 (S. Li et al., 2018), LSTMs have convergence issues when training long sequences (length $\geqslant$ 1000). Because of this, we considered that the results of the bi-LSTM were not comparable to those from the other architectures and thus, decided to remove it from the analysis.

The description of the metrics used for evaluating and comparing the performance of the different padding types is shown below. Let TP be the number of true positive classified samples, TN the true negatives, FP the false positives and FN the false negatives:

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

.

If $precision = \frac{TP}{TP+FP}$ and $recall = \frac{TP}{TP+FN}$, then F1-score can be described as

$$F_1 = 2 \cdot \frac{precision \cdot recall}{precision + recall}$$

The macro F1-score calculates metrics for each label, and finds their unweighted mean. This does not take label imbalance into account.

The Area Under the Curve (AUC) of the Receiver Operating Characteristic (ROC) curve measures performance in classification problems for different thresholds. The ROC curve is a probability curve obtained by plotting the True Positive Rate (which is the same as the recall) on y-axis against the False Positive Rate (which is $\frac{FP}{TN+FP}$) on the x-axis. The AUC, which is the area under this curve, quantifies how capable the model is of distinguishing both classes.

C DATA IMBALANCE

BALANCING DATA ON DEEP LEARNING–BASED PROTEOCHEMOMETRIC ACTIVITY CLASSIFICATION

C.1 APPENDIX S1 – DATA IMBALANCE

C.1.1 Materials and methods

Table S29 summarizes the GPCRs, nuclear receptors and proteases subdatasets, used for validating the results obtained in the kinases family.

	GPCR	NR	PR
Entity		#	
Compounds	120,906	18,120	58,882
Targets	498	82	319
Ligand-target pairs	200,523	26,433	92,965
Actives	153,993	16,962	57,510
Inactives	46,530	9,471	35,455

Table S29: Summary of the GPCRs, nuclear receptors (NR) and proteases (PR) subdatasets.

On the other hand, table S30 displays the number of active and inactive protein-compound pairs, averaged over folds, that were available in each set (training, validation, test) per strategy and protein family.

In Figure S72, an histogram with the proportion of actives respect to all the interactions per protein is shown for kinases, GPCRs, nuclear receptors and proteases. It can be seen that for both kinases GPCRs (but specially for GPCRs) there are more actives per protein than inactives. There is a large number of GPCR proteins for which only active interactions are reported. For proteases the distribution is more even.

In Figure S73, the number of active and inactive interactions for each protein is represented for kinases, GPCRs, nuclear receptors and proteases. In all cases, there are many proteins for which there are only active or inactive interactions, thus the median is near 0. There are also many proteins (shown as outliers) with a large number of active interactions.

As stated in the main text, proteins whose sample size in a specific fold or strategy did not allow SMOTE upsampling were excluded. Table S31

This appendix reproduces the supplementary data of: Angela Lopez-del Rio, Sergio-Picart Armada and Alexandre Perera-Lluna. "Balancing Data on Proteochemometrics Activity Classification". *ChemRxiv (2021)*.

Table S30: Total number of protein-compound pairs in each strategy and protein family, displayed in the actives/inactives format. Number of actives and inactives were averaged over the 10 folds.

protein_type	strategy	training	validation	test
kinases	no_resampling	75767/23511	10970/3429	10215/3045
kinases	resampling_after_clustering	76897/76897	9778/9778	8955/8955
kinases	resampling_before_clustering	78102/77361	9899/10456	10374/10558
kinases	semi_resampling	77951/77539	9893/10305	10215/3045
GPCRs	no_resampling	121449/36876	16315/4919	16228/4734
GPCRs	resampling_after_clustering	121097/121097	14355/14355	13930/13930
GPCRs	resampling_before_clustering	121108/120957	15613/15307	17235/17692
GPCRs	semi_resampling	122132/120912	15487/16708	16228/4734
nuclear_receptors	no_resampling	13333/7479	1824/997	1804/994
nuclear_receptors	resampling_after_clustering	14218/14218	1760/1760	1678/1678
nuclear_receptors	resampling_before_clustering	14123/14285	1912/1856	2074/1966
nuclear_receptors	semi_resampling	14372/14398	1798/1771	1804/994
proteases	no_resampling	44954/28098	6314/3586	6243/3770
proteases	resampling_after_clustering	50558/50558	6323/6323	6229/6229
proteases	resampling_before_clustering	50891/51481	6766/6778	7217/6616
proteases	semi_resampling	51091/51221	6604/6474	6243/3770

shows the number of protein kinases for which performance metrics were computed in each case.

Table S31: Number of proteins for which performance metrics were computed. The resampling after clustering was the most stringent strategy regarding eligible proteins, since the resampling was carried out after the clustering, which introduced more imbalance.

Strategy	Fold 0	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Fold 6	Fold 7	Fold 8	Fold 9
no_resampling	288	282	295	303	305	305	293	307	294	301
resampling_before_clustering	271	295	284	274	298	286	301	307	302	293
semi_resampling	288	282	295	303	305	305	293	307	294	301
resampling_after_clustering	79	74	72	84	76	79	87	82	81	100

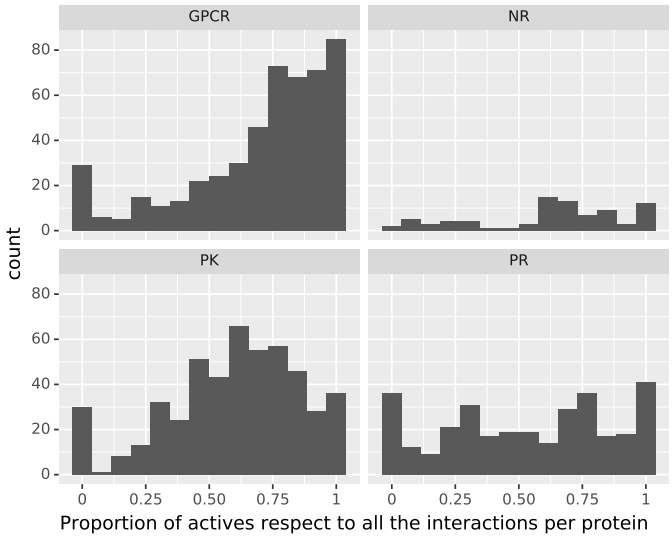

Figure S72: Histogram of proportion of actives respect to the total number of interactions for each protein for the protein kinases (PK), GPCRs, nuclear receptors (NR) and proteases (PR) families.

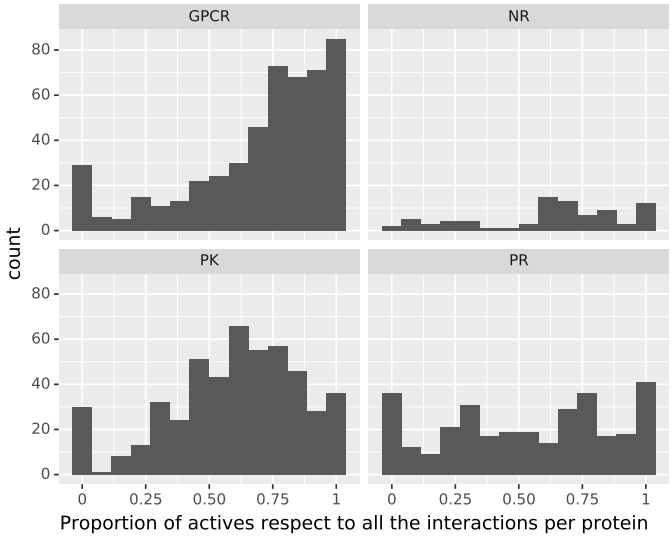

Figure S73: Histogram of proportion of actives respect to the total number of interactions for each protein for the protein kinases (PK), GPCRs, nuclear receptors (NR) and proteases (PR) families.

C.1.2 Model architecture

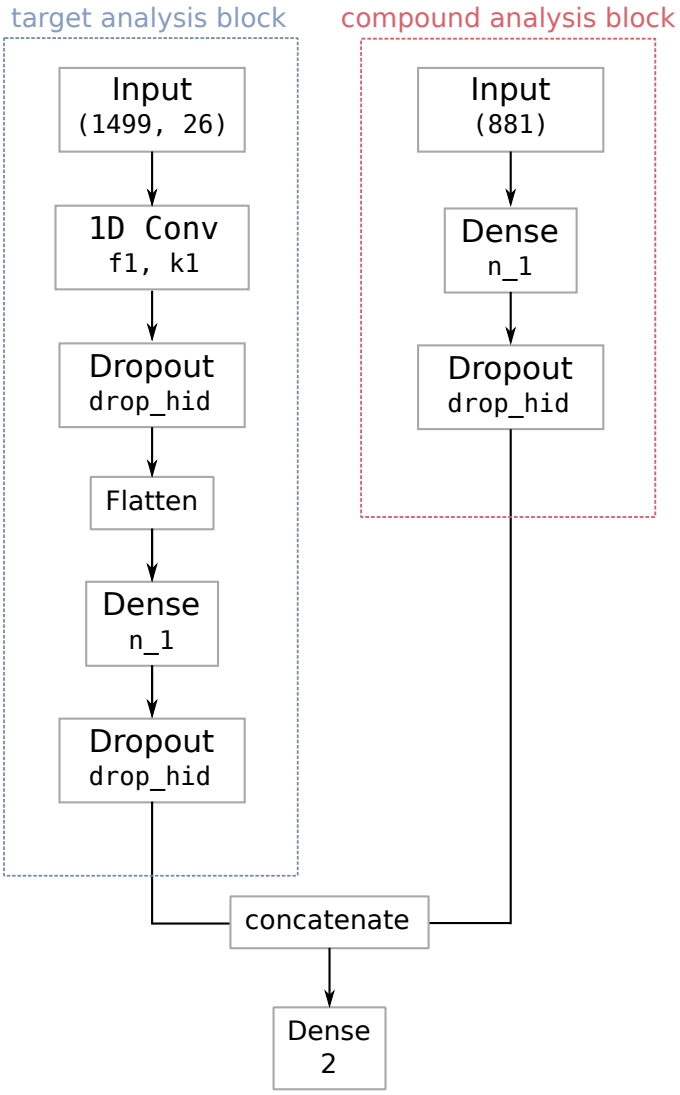

Figure S74: Schematic representation of the deep learning architecture. It is composed of two main blocks: the protein analysis block and the compound analysis block.

The schematic representation of the deep learning model is displayed in Figure S74. The selected parameters were drop_hid=0.4, n_1=50, f1=64 and k1=3 . As activation function, we used Rectified Linear Unit (ReLU) for the hidden layers (LeCun et al., 2015), and *softmax* for the output dense (2) layer.

C.1.3 Performance metrics

The description of the metrics used for evaluating and comparing the performance of the different balancing strategies is shown below. Let TP be the number of true positive classified samples, TN the true negatives, FP the false positives and FN the false negatives:

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

.

Since accuracy is not a proper metric when the dataset is highly imbalanced, we used balanced accuracy, which is the average between sensitivity and specificity:

$$Balanced\ accuracy = \left[\frac{TP}{TP + FN} + \frac{TN}{TN + FP}\right]/2$$

If $precision = \frac{TP}{TP+FP}$ and $recall = \frac{TP}{TP+FN}$, then F1-score can be described as

$$F_1 = 2 \cdot \frac{precision \cdot recall}{precision + recall}$$

The macro F1-score calculates metrics for each label, and finds their unweighted mean. This does not take label imbalance into account.

The Matthews correlation coefficient (MCC) is alue between -1 and +1: +1 represents a perfect prediction, 0 an average random prediction and -1 an inverse prediction. This metric is not sensitive to imbalance, so it can be used even if the classes are of very different sizes. It can be calculated using the formula:

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

.

The Area Under the Curve (AUC) of the Receiver Operating Characteristic (ROC) curve measures performance in classification problems for different thresholds. The ROC curve is a probability curve obtained by plotting the True Positive Rate (which is the same as the recall) on y-axis against the False Positive Rate (which is $\frac{FP}{TN+FP}$) on the x-axis. The AUC, which is the area under this curve, quantifies how capable the model is of distinguishing both classes.

C.1.4 Results

Description of data balance

DISTRIBUTIONS OF ACTIVES RATIO In Figure S75, the distributions of the proportion of active molecules for a protein for each strategy both on training and test sets is shown.

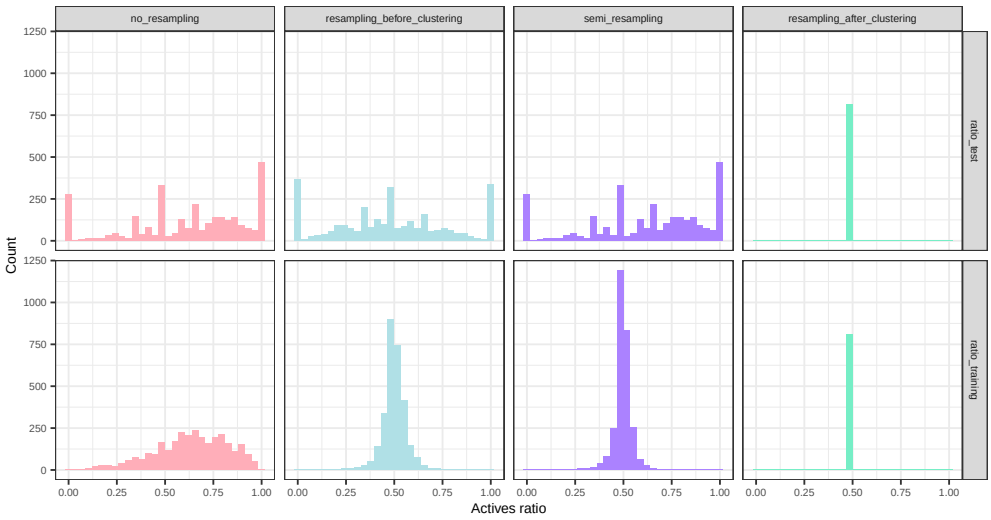

Figure S75: Distribution of the actives ratio in the training set and in the test set.

TRAINING AND TEST IMBALANCE COMPARISON Table S32 shows the Pearson correlation estimate, 95% interval and p-value for each strategy (except *resampling after clustering*, where ratios are constant).

Table S32: Correlations between train and test active ratios. 95% confidence intervals and p-values are shown.

strategy	cor	ci_lower	ci_upper	alternative	pvalue
no_resampling	0.369	0.338	0.400	two.sided	1.15e-96
resampling_before_clustering	-0.428	-0.457	-0.398	two.sided	5.54e-130
semi_resampling	0.014	-0.024	0.051	two.sided	4.76e-01
resampling_after_clustering	NA	NA	NA	two.sided	NA

OTHER COVARIATES Figure S76 shows the effect of the number of interactions of each protein in its corresponding set and fold on the test set imbalance. Figure S77 shows the effect of the protein length on the test set imbalance.

Pearson correlation estimates, 95% interval an p-value between imbalance and number of interactions for each strategy is shown in Table S33. The same information but for the correlations between imbalance and sequence length is in Table S34.

Table S33: Correlations between imbalance (as defined above) and number of interactions. 95% confidence intervals and p-values are shown.

strategy	cor	ci_lower	ci_upper	alternative	pvalue
no_resampling	-0.061	-0.097	-0.026	two.sided	8.01e-04
resampling_before_clustering	-0.274	-0.307	-0.240	two.sided	3.98e-51
semi_resampling	-0.061	-0.097	-0.026	two.sided	8.01e-04
resampling_after_clustering	NA	NA	NA	two.sided	NA

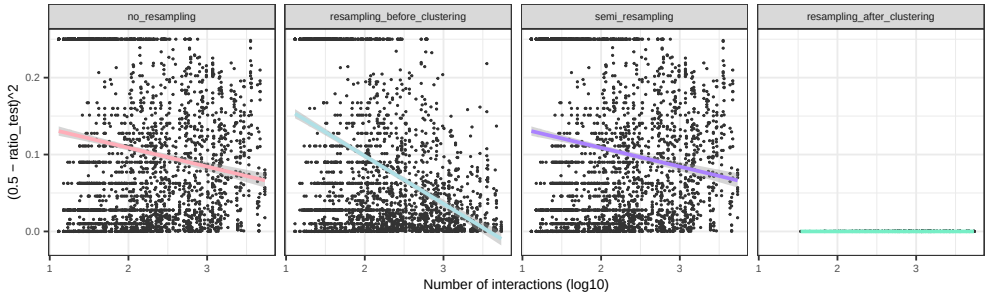

Figure S76: Data imbalance in the test set as a function of the number of available interactions for each protein. The imbalance was quantified as $(0.5 - \texttt{ratio}_{\texttt{test}})^2$ in order to measure the deviation from a perfectly balanced protein, with an actives ratio of 0.5.

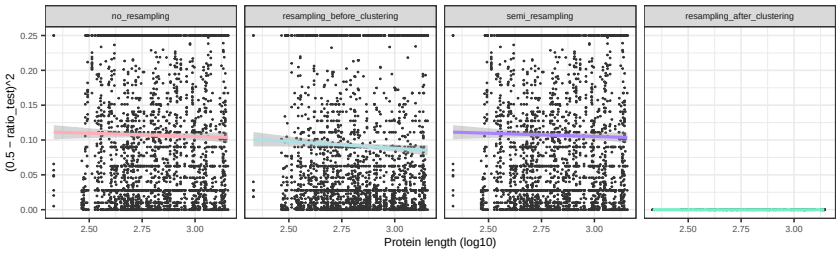

Figure S77: Data imbalance in the test set as a function of the protein length, in amino acids.

Table S34: Correlations between imbalance (as defined above) and sequence length. 95% confidence intervals and p-values are shown.

strategy	cor	ci_lower	ci_upper	alternative	pvalue
no_resampling	-0.016	-0.052	0.020	two.sided	3.73e-01
resampling_before_clustering	-0.046	-0.082	-0.009	two.sided	1.37e-02
semi_resampling	-0.016	-0.052	0.020	two.sided	3.73e-01
resampling_after_clustering	NA	NA	NA	two.sided	NA

Analysis of the predicted proportions

Pearson correlation estimates, 95% interval an p-value between training and predicted test active ratios for each strategy is shown in Table S36.

In Table S35 it is shown the percentage of extreme cases for each strategy.

The quasibinomial generalized linear models describing the predicted active ratio are depicted in Table S37 for the *semi resampling* and *resampling before clustering* strategy and in Table S38 for the *no resampling*.

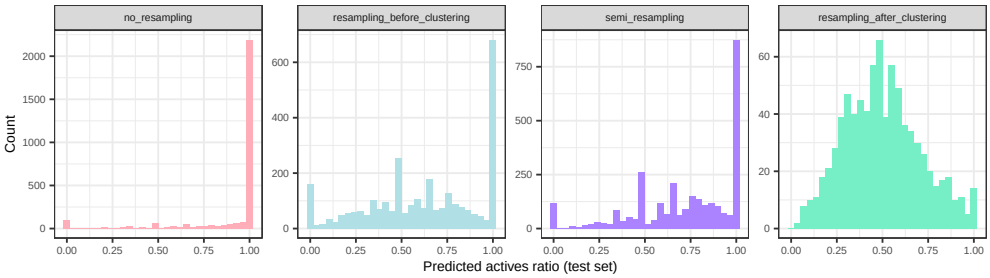

Figure S78: Ratios of the prediction values, after binarization.

Table S35: Percentage of extreme cases, i.e. proteins with all actives or inactives.

strategy	RatioSet	all_inactives	all_actives	all_extremes
no_resampling	ratio_test	9.2	15.5	24.7
no_resampling	ratio_test_predicted	3.5	71.6	75.1
no_resampling	ratio_training	0.0	0.0	0.0
resampling_before_clustering	ratio_test	12.5	11.5	24.0
resampling_before_clustering	ratio_test_predicted	5.5	23.4	28.9
resampling_before_clustering	ratio_training	0.0	0.0	0.0
semi_resampling	ratio_test	9.2	15.5	24.7
semi_resampling	ratio_test_predicted	4.0	29.1	33.1
semi_resampling	ratio_training	0.0	0.0	0.0
resampling_after_clustering	ratio_test	0.0	0.0	0.0
resampling_after_clustering	ratio_test_predicted	0.0	1.2	1.2
resampling_after_clustering	ratio_training	0.0	0.0	0.0

Table S36: Correlations between train and predicted test active ratios. 95% confidence intervals and p-values are shown.

strategy	cor	ci_lower	ci_upper	alternative	pvalue
no_resampling	0.469	0.440	0.496	two.sided	2.79e-162
resampling_before_clustering	-0.094	-0.130	-0.058	two.sided	3.77e-07
semi_resampling	0.008	-0.029	0.045	two.sided	6.78e-01
resampling_after_clustering	NA	NA	NA	two.sided	NA

Table S37: Explanatory models to describe the predicted active ratio for the *semi resampling* and the *resampling before clustering* strategies. Significance and 95% confidence intervals are included.

	semi_resampling	resampling_before_clustering
	(1)	(2)
ratio_training	0.197 (−0.903, 1.298)	−0.446 (−1.296, 0.405)
	p = 0.725	p = 0.305
ratio_test	0.945 (0.775, 1.114)***	0.784 (0.606, 0.963)***
	p = 2.460e-27	p = 1.181e-17
log10(n_interactions)	−0.391 (−0.467, −0.314)***	−0.396 (−0.466, −0.325)***
	p = 3.197e-23	p = 1.987e-27
log10(len_seq)	0.289 (0.023, 0.554)*	−0.033 (−0.293, 0.226)
	p = 0.033	p = 0.801
fold1	0.034 (−0.176, 0.245)	0.071 (−0.138, 0.281)
	p = 0.748	p = 0.504
fold2	−0.651 (−0.852, −0.45)***	0.416 (0.196, 0.635)**
	p = 2.348e-10	p = 2.076e-04
fold3	0.982 (0.74, 1.224)***	−0.436 (−0.646, −0.226)**
	p = 2.598e-15	p = 4.738e-05
fold4	0.665 (0.44, 0.891)***	0.326 (0.114, 0.538)*
	p = 8.708e-09	p = 2.609e-03
fold5	0.023 (−0.187, 0.232)	0.333 (0.118, 0.548)*
	p = 0.831	p = 2.413e-03
fold6	−0.524 (−0.725, −0.323)***	0.021 (−0.186, 0.229)
	p = 3.457e-07	p = 0.839
fold7	0.626 (0.402, 0.849)***	0.377 (0.165, 0.589)**
	p = 4.506e-08	p = 5.045e-04
fold8	0.504 (0.283, 0.725)**	0.378 (0.165, 0.592)**
	p = 8.130e-06	p = 5.250e-04
fold9	−8.292e-03 (−0.215, 0.199)	−0.73 (−0.938, −0.522)***
	p = 0.937	p = 7.301e-12
Constant	0.177 (−0.76, 1.113)	1.229 (0.34, 2.118)*
	p = 0.712	p = 6.796e-03
Observations	2783	2911

Note: *p<0.05; **p<1.000e-03; ***p<1e-06

Table S38: Explanatory models to describe the predicted active ratio for the *no resampling* strategy. Significance and 95% confidence intervals are included.

	no_resampling
ratio_training	8.312 (7.581, 9.042)***
	p = 0.000e+00
ratio_test	1.102 (0.741, 1.464)***
	p = 2.600e-09
log10(n_interactions)	−1.24 (−1.422, −1.058)***
	p = 1.330e-39
log10(len_seq)	−0.949 (−1.513, −0.386)**
	p = 9.683e-04
fold1	−0.505 (−0.931, −0.078)*
	p = 0.021
fold2	−0.034 (−0.478, 0.411)
	p = 0.881
fold3	1.063 (0.545, 1.58)**
	p = 5.828e-05
fold4	−0.805 (−1.21, −0.4)**
	p = 9.970e-05
fold5	0.917 (0.411, 1.423)**
	p = 3.862e-04
fold6	1.709 (1.097, 2.321)***
	p = 4.711e-08
fold7	0.169 (−0.278, 0.615)
	p = 0.459
fold8	0.116 (−0.341, 0.574)
	p = 0.618
fold9	0.886 (0.385, 1.387)**
	p = 5.365e-04
Constant	1.904 (0.269, 3.539)*
	p = 0.023
Observations	2973
Note:	*p<0.05; **p<1.000e-03; ***p<1e-06

Performance metrics

BASELINE PERFORMANCE Figure S79 shows a fold-averaged picture of the metrics by protein.

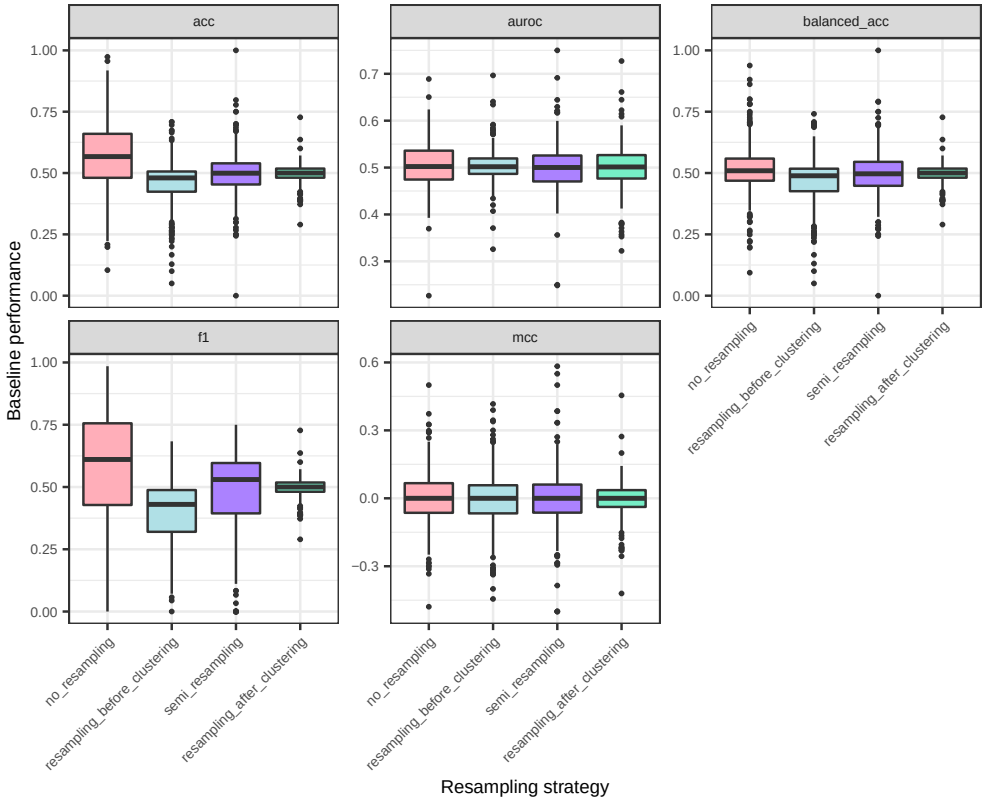

Figure S79: Performance metrics for imbalance-aware random baselines. Data points correspond to proteins, averaged over folds.

The quartiles for the baseline F1-scores are in Table S39.

Table S39: Quartiles for the baseline F1-scores.

strategy	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
no_resampling	0.000	0.428	0.611	0.569	0.756	0.985
resampling_before_clustering	0.000	0.320	0.430	0.392	0.488	0.683
semi_resampling	0.000	0.394	0.530	0.478	0.596	0.750
resampling_after_clustering	0.290	0.481	0.500	0.497	0.518	0.727

Table S40 shows the type 3 ANOVA p-values of the strategy variable in the explanatory linear model of Equation 2 (main text) applied to the random baseline metrics.

Table S40: ANOVA p-values for including the resampling strategy as a regressor. Significant p-values imply that differences exist between resampling strategies.

strategy	variable	Sum Sq	Df	F value	Pr(>F)
acc	strategy	22.2816465	3	1.36e+02	5.58e-86
auroc	strategy	0.0324079	3	1.79e-01	9.11e-01
balanced_acc	strategy	3.0761723	3	1.70e+01	5.61e-11
f1	strategy	53.2175335	3	2.36e+02	8.97e-148
mcc	strategy	0.1219509	3	3.03e-01	8.24e-01

DEEP LEARNING MODEL Table S41 summarizes the number of proteins, added over folds, whose metrics were computable.

Table S41: Number of computable performance measures. AUROC was undefined for proteins with all actives or unactives in the test set, hence its lower counts.

strategy	acc	auroc	f1	balanced_acc	mcc
no_resampling	2973	2238	2973	2973	2973
resampling_before_clustering	2911	2211	2911	2911	2911
semi_resampling	2973	2238	2973	2973	2973
resampling_after_clustering	814	814	814	814	814

Absolute, baseline-naive performance The strategy covariate was always significant in a type 3 ANOVA (table S42). The explanatory linear models are summarized in S43.

Table S42: ANOVA p-values for including the resampling strategy as a regressor in the performance models.

strategy	variable	Sum Sq	Df	F value	Pr(>F)
acc	strategy	5.170321	3	2.37e+01	2.89e-15
auroc	strategy	10.908150	3	4.78e+01	1.28e-30
f1	strategy	29.754722	3	1.09e+02	2.07e-69
balanced_acc	strategy	5.464791	3	2.72e+01	1.64e-17
mcc	strategy	48.559151	3	1.82e+02	7.97e-115

Pairwise comparisons of the strategy coefficients using Tukey's method are shown in Figure S80.

Table S44 shows the expected absolute performances by metric and strategy.

Table S43: Linear models to describe each performance metric. Standard deviations in parentheses.

	acc	auroc	f1	balanced_acc	mcc
	(1)	(2)	(3)	(4)	(5)
strategyresampling_before_clustering	−0.046***	0.084***	−0.132***	0.047***	0.148***
	(7.036e-03)	(8.279e-03)	(7.867e-03)	(6.749e-03)	(7.777e-03)
	p = 8.802e-11	p = 4.087e-24	p = 1.533e-62	p = 3.584e-12	p = 1.229e-79
strategysemi_resampling	−0.046***	1.799e-03	−0.066***	8.309e-03	0.041***
	(6.997e-03)	(8.241e-03)	(7.823e-03)	(6.712e-03)	(7.734e-03)
	p = 3.308e-11	p = 0.827	p = 3.896e-17	p = 0.216	p = 1.524e-07
strategyresampling_after_clustering	−0.069***	0.061***	−0.144***	0.07***	0.209***
	(0.011)	(0.012)	(0.013)	(0.011)	(0.012)
	p = 7.285e-10	p = 1.668e-07	p = 1.727e-30	p = 6.696e-11	p = 6.559e-63
log10(n_interactions)	0.098***	0.076***	0.123***	0.041***	0.114***
	(4.212e-03)	(4.958e-03)	(4.709e-03)	(4.040e-03)	(4.655e-03)
	p = 0.000e+00	p = 6.946e-52	p = 0.000e+00	p = 2.529e-24	p = 0.000e+00
log10(len_seq)	0.052**	0.024	0.053*	0.028*	0.043*
	(0.015)	(0.017)	(0.017)	(0.014)	(0.017)
	p = 5.105e-04	p = 0.157	p = 1.612e-03	p = 0.05	p = 9.744e-03
fold1	0.041*	−7.487e-03	0.036*	0.033*	−0.025
	(0.013)	(0.015)	(0.014)	(0.012)	(0.014)
	p = 1.037e-03	p = 0.617	p = 9.481e-03	p = 5.917e-03	p = 0.07
fold2	0.031*	−0.014	0.035*	0.018	−0.022
	(0.012)	(0.015)	(0.014)	(0.012)	(0.014)
	p = 0.014	p = 0.33	p = 0.012	p = 0.13	p = 0.103
fold3	0.057**	1.904e-03	0.073***	0.033*	1.973e-04
	(0.012)	(0.014)	(0.014)	(0.012)	(0.014)
	p = 5.280e-06	p = 0.896	p = 1.650e-07	p = 5.870e-03	p = 0.989
fold4	4.646e-03	−5.580e-03	0.023	−0.01	−7.461e-03
	(0.012)	(0.014)	(0.014)	(0.012)	(0.014)
	p = 0.707	p = 0.699	p = 0.097	p = 0.384	p = 0.585
fold5	0.035*	0.053**	0.081***	0.011	0.03*
	(0.012)	(0.014)	(0.014)	(0.012)	(0.014)
	p = 4.349e-03	p = 1.862e-04	p = 5.174e-09	p = 0.345	p = 0.03
fold6	0.016	−0.011	0.018	9.628e-03	6.239e-03
	(0.012)	(0.015)	(0.014)	(0.012)	(0.014)
	p = 0.208	p = 0.455	p = 0.182	p = 0.418	p = 0.648
fold7	0.027*	0.036*	0.066**	0.015	−2.417e-03
	(0.012)	(0.014)	(0.014)	(0.012)	(0.014)
	p = 0.031	p = 0.011	p = 1.775e-06	p = 0.198	p = 0.859
fold8	0.021	1.533e-03	0.056**	−1.289e-03	−0.024
	(0.012)	(0.014)	(0.014)	(0.012)	(0.014)
	p = 0.096	p = 0.915	p = 5.076e-05	p = 0.914	p = 0.085
fold9	0.025*	0.015	0.021	0.014	0.018
	(0.012)	(0.014)	(0.014)	(0.012)	(0.014)
	p = 0.045	p = 0.307	p = 0.134	p = 0.244	p = 0.193
Constant	0.274***	0.347***	0.243**	0.378***	−0.318***
	(0.045)	(0.051)	(0.05)	(0.043)	(0.049)
	p = 7.742e-10	p = 1.421e-11	p = 1.039e-06	p = 1.020e-18	p = 1.102e-10
Observations	9671	7501	9671	9671	9671
R^2	0.063	0.059	0.096	0.027	0.134
Adjusted R^2	0.062	0.057	0.095	0.026	0.133

Note: *p<0.05; **p<1.000e-03; ***p<1e-06

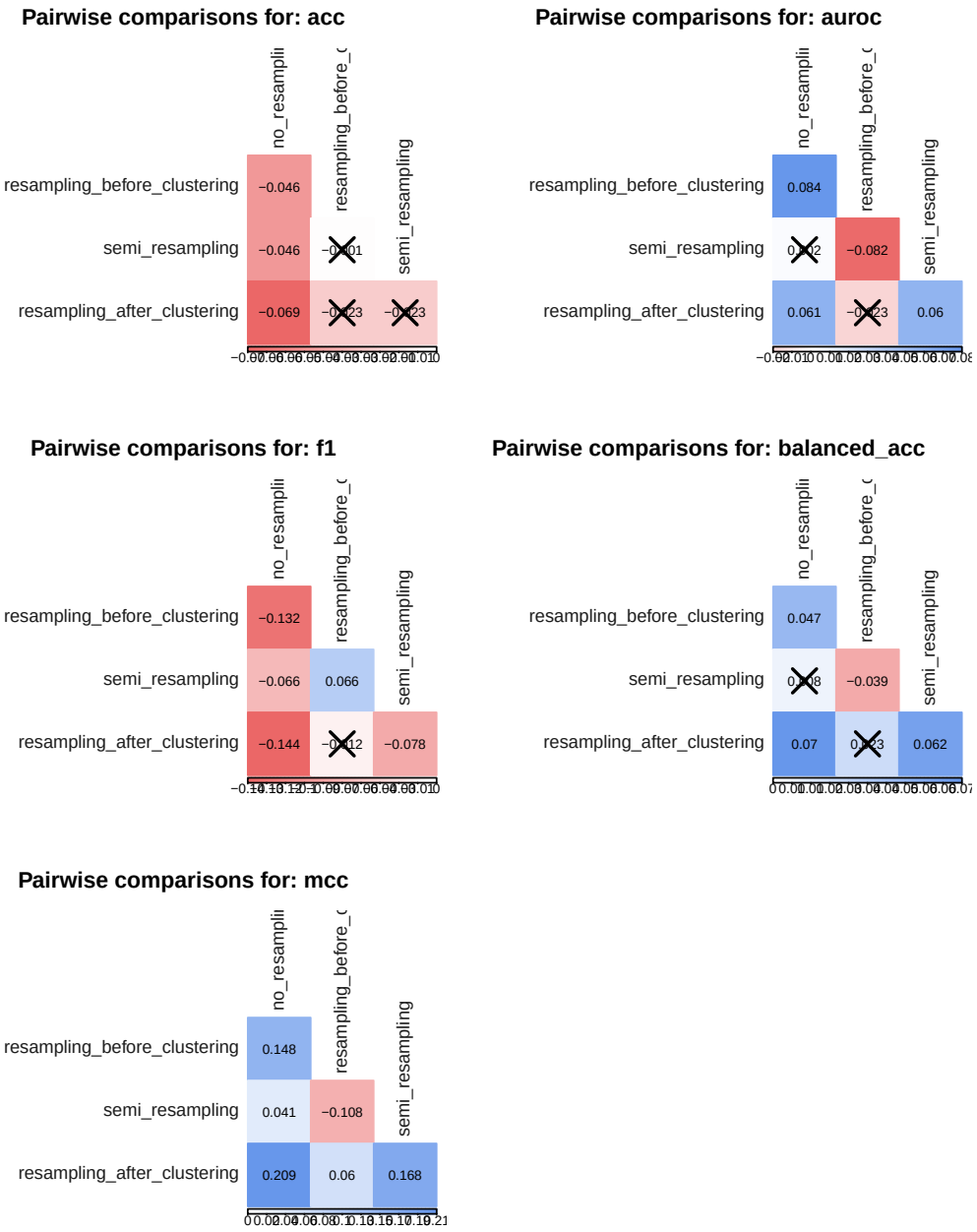

Figure S80: Pairwise comparison of strategy performance using Tukey method.

Table S44: Expected absolute performances, by metric and strategy, with 95% confidence intervals.

metric	strategy	emmean	SE	lower.CL	upper.CL
acc	no_resampling	0.712	5.614e-03	0.701	0.723
acc	resampling_before_clustering	0.666	5.618e-03	0.655	0.677
acc	semi_resampling	0.665	5.614e-03	0.654	0.676
acc	resampling_after_clustering	0.643	9.511e-03	0.624	0.662
auroc	no_resampling	0.627	6.280e-03	0.615	0.640
auroc	resampling_before_clustering	0.712	6.257e-03	0.699	0.724
auroc	semi_resampling	0.629	6.280e-03	0.617	0.641
auroc	resampling_after_clustering	0.689	9.737e-03	0.670	0.708
f1	no_resampling	0.766	6.277e-03	0.754	0.779
f1	resampling_before_clustering	0.634	6.282e-03	0.622	0.646
f1	semi_resampling	0.700	6.277e-03	0.688	0.713
f1	resampling_after_clustering	0.622	1.063e-02	0.601	0.643
balanced_acc	no_resampling	0.582	5.385e-03	0.572	0.593
balanced_acc	resampling_before_clustering	0.629	5.389e-03	0.619	0.640
balanced_acc	semi_resampling	0.591	5.385e-03	0.580	0.601
balanced_acc	resampling_after_clustering	0.652	9.123e-03	0.634	0.670
mcc	no_resampling	0.108	6.205e-03	0.096	0.120
mcc	resampling_before_clustering	0.256	6.210e-03	0.244	0.268
mcc	semi_resampling	0.149	6.205e-03	0.136	0.161
mcc	resampling_after_clustering	0.317	1.051e-02	0.296	0.337

Baseline-adjusted performance In this case, the strategy covariate was also always significant in a type 3 ANOVA (Table S45). The explanatory linear models for the adjusted metrics are summarized in S46.

Pairwise comparisons of the strategy coefficients using Tukey's method for the adjusted metrics are shown in Figure S81.

Expected adjusted performances by metric and strategy are in Table S47.

Table S45: ANOVA p-values for including the resampling strategy as a regressor in the adjusted performance models.

strategy	variable	Sum Sq	Df	F value	Pr(>F)
acc	strategy	7.228605	3	2.09e+01	1.79e-13
auroc	strategy	12.123911	3	2.96e+01	4.81e-19
f1	strategy	3.665732	3	1.43e+01	2.78e-09
balanced_acc	strategy	15.297979	3	4.28e+01	1.82e-27
mcc	strategy	52.471976	3	7.77e+01	1.29e-49

Table S46: Linear models to describe each adjusted performance metric. Standard deviations in parentheses.

	acc	auroc	f1	balanced_acc	mcc
	(1)	(2)	(3)	(4)	(5)
strategyresampling_before_clustering	0.07***	0.087***	0.05***	0.093***	0.157***
	(8.861e-03)	(0.011)	(7.628e-03)	(9.002e-03)	(0.012)
	p = 3.431e-15	p = 3.742e-15	p = 6.971e-11	p = 1.129e-24	p = 1.213e-36
strategysemi_resampling	0.037**	−2.411e-03	0.026**	0.029*	0.041*
	(8.966e-03)	(0.011)	(7.719e-03)	(9.108e-03)	(0.013)
	p = 4.267e-05	p = 0.829	p = 7.937e-04	p = 1.332e-03	p = 1.190e-03
strategyresampling_after_clustering	0.043*	0.06**	0.03*	0.098***	0.209***
	(0.014)	(0.016)	(0.012)	(0.014)	(0.02)
	p = 2.204e-03	p = 1.412e-04	p = 0.013	p = 9.770e-12	p = 5.834e-26
log10(n_interactions)	0.053***	0.079***	4.567e-03	0.026**	0.118***
	(5.390e-03)	(6.740e-03)	(4.640e-03)	(5.475e-03)	(7.529e-03)
	p = 2.346e-22	p = 3.209e-31	p = 0.325	p = 2.408e-06	p = 9.075e-55
log10(len_seq)	0.046*	0.018	0.021	0.02	0.025
	(0.019)	(0.023)	(0.016)	(0.019)	(0.027)
	p = 0.016	p = 0.439	p = 0.2	p = 0.294	p = 0.35
fold1	0.029	−6.058e-03	7.826e-03	0.021	−0.023
	(0.016)	(0.02)	(0.014)	(0.016)	(0.022)
	p = 0.071	p = 0.763	p = 0.567	p = 0.183	p = 0.296
fold2	0.024	−0.026	−8.380e-03	0.014	−0.044*
	(0.016)	(0.02)	(0.014)	(0.016)	(0.022)
	p = 0.126	p = 0.193	p = 0.539	p = 0.388	p = 0.046
fold3	0.049*	−3.319e-03	0.027*	0.029	−0.022
	(0.016)	(0.02)	(0.014)	(0.016)	(0.022)
	p = 1.775e-03	p = 0.865	p = 0.046	p = 0.073	p = 0.317
fold4	9.957e-03	1.882e-03	0.013	−7.284e-04	3.653e-03
	(0.016)	(0.019)	(0.014)	(0.016)	(0.022)
	p = 0.527	p = 0.923	p = 0.343	p = 0.964	p = 0.868
fold5	0.024	0.057*	0.024	9.004e-03	0.025
	(0.016)	(0.019)	(0.014)	(0.016)	(0.022)
	p = 0.134	p = 3.174e-03	p = 0.079	p = 0.575	p = 0.255
fold6	2.762e-03	−0.023	3.872e-03	7.173e-04	−6.218e-03
	(0.016)	(0.02)	(0.014)	(0.016)	(0.022)
	p = 0.861	p = 0.248	p = 0.775	p = 0.964	p = 0.777
fold7	0.022	0.032	0.033*	0.013	−0.011
	(0.016)	(0.019)	(0.013)	(0.016)	(0.022)
	p = 0.159	p = 0.099	p = 0.015	p = 0.406	p = 0.601
fold8	7.448e-03	−3.576e-04	5.280e-03	−0.012	−0.033
	(0.016)	(0.019)	(0.014)	(0.016)	(0.022)
	p = 0.636	p = 0.985	p = 0.697	p = 0.457	p = 0.132
fold9	0.019	7.796e-03	−9.248e-03	9.900e-03	−2.422e-03
	(0.016)	(0.019)	(0.013)	(0.016)	(0.022)
	p = 0.216	p = 0.684	p = 0.493	p = 0.534	p = 0.912
Constant	−0.184*	−0.141*	0.03	−0.08	−0.273**
	(0.057)	(0.069)	(0.049)	(0.057)	(0.079)
	p = 1.143e-03	p = 0.041	p = 0.534	p = 0.166	p = 5.493e-04
Observations	9473	7387	9473	9473	9473
R^2	0.02	0.036	7.025e-03	0.019	0.061
Adjusted R^2	0.019	0.035	5.555e-03	0.017	0.06

Note: *p<0.05; **p<1.000e-03; ***p<1e-06

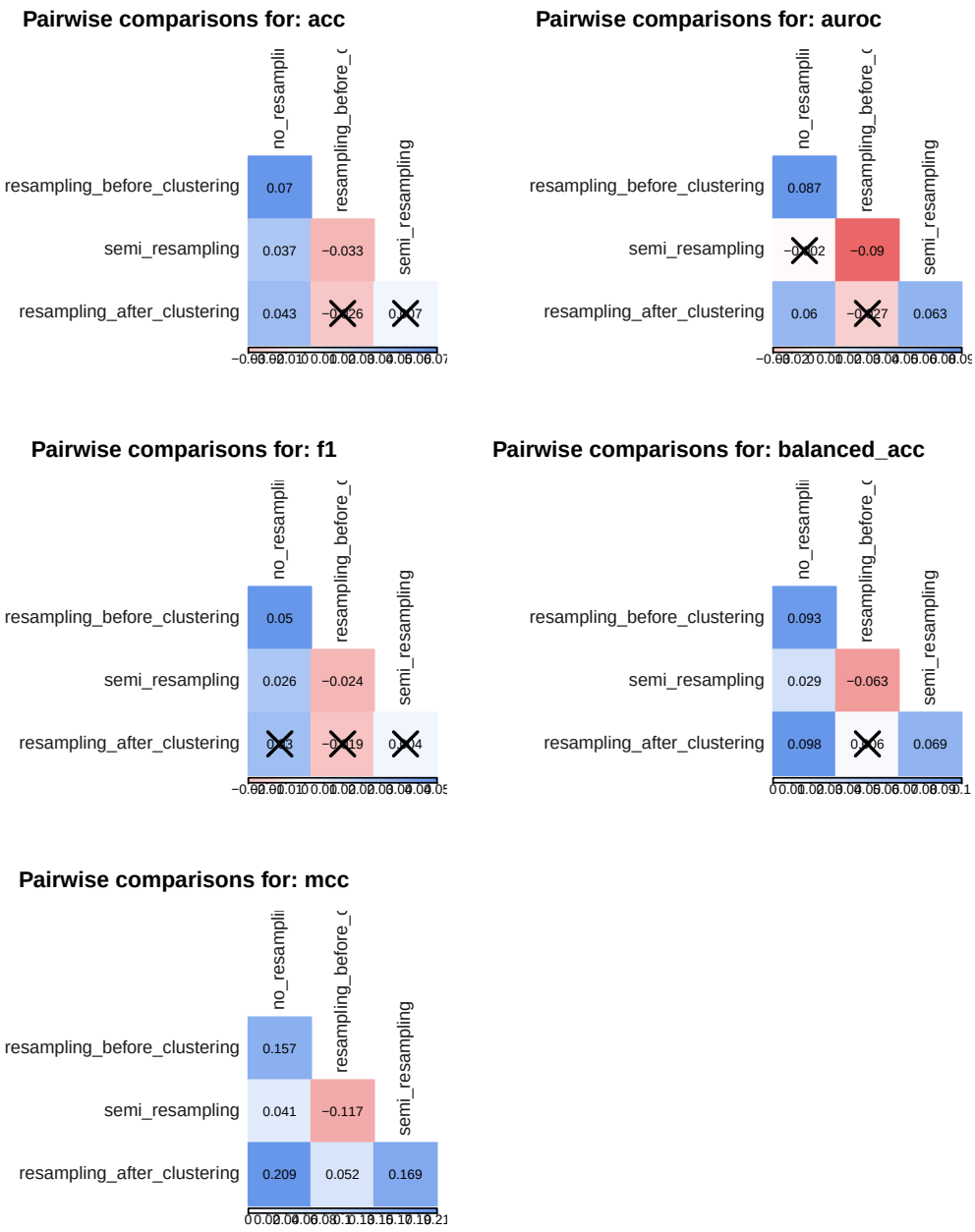

Figure S81: Pairwise comparison of strategy adjusted performance using Tukey method.

Table S47: Expected adjusted performances, by metric and strategy, with 95% confidence intervals.

metric	strategy	emmean	SE	lower.CL	upper.CL
acc	no_resampling	0.108	7.116e-03	0.094	0.122
acc	resampling_before_clustering	0.177	7.119e-03	0.164	0.191
acc	semi_resampling	0.144	7.180e-03	0.130	0.158
acc	resampling_after_clustering	0.151	1.204e-02	0.127	0.175
auroc	no_resampling	0.127	8.448e-03	0.110	0.144
auroc	resampling_before_clustering	0.214	8.413e-03	0.198	0.231
auroc	semi_resampling	0.125	8.528e-03	0.108	0.141
auroc	resampling_after_clustering	0.187	1.311e-02	0.161	0.213
f1	no_resampling	0.112	6.126e-03	0.100	0.124
f1	resampling_before_clustering	0.162	6.128e-03	0.150	0.174
f1	semi_resampling	0.138	6.181e-03	0.126	0.150
f1	resampling_after_clustering	0.143	1.036e-02	0.122	0.163
balanced_acc	no_resampling	0.057	7.229e-03	0.043	0.071
balanced_acc	resampling_before_clustering	0.149	7.232e-03	0.135	0.163
balanced_acc	semi_resampling	0.086	7.294e-03	0.072	0.100
balanced_acc	resampling_after_clustering	0.155	1.223e-02	0.131	0.179
mcc	no_resampling	0.106	9.940e-03	0.086	0.125
mcc	resampling_before_clustering	0.263	9.944e-03	0.243	0.282
mcc	semi_resampling	0.146	1.003e-02	0.127	0.166
mcc	resampling_after_clustering	0.315	1.682e-02	0.282	0.348

Further validation for other protein families

Table S48 shows the correlations between training and sets sets active ratios for GPCRs.

Table S48: Correlations between train and test active ratios. 95% confidence intervals and p-values are shown.

strategy	cor	ci_lower	ci_upper	alternative	pvalue
no_resampling	0.594	0.572	0.616	two.sided	6.73e-320
resampling_before_clustering	-0.386	-0.417	-0.354	two.sided	4.80e-98
semi_resampling	0.067	0.029	0.105	two.sided	5.91e-04
resampling_after_clustering	NA	NA	NA	two.sided	NA

Table S49 shows the linear models describing the predicted active ratio for the *semi resampling* and the *resampling after clustering* strategies.

Table S50 summarizes the linear models describing each adjusted performance metric.

C.2 APPENDIX S2 – MODEL PREDICTIONS AND PERFORMANCE (GPCRS)

C.2.1 Overview

This supplement describes the behaviour of the proteochemometrics (PCM) deep learning model to predict protein-compound bioactivity for **GPCRs**. Specifically, this includes the descriptive statistics of data imbalance: the pro-

Table S49: Linear models to describe the predicted active ratio for the *semi resampling* and the *resampling before clustering* strategies. Significance and 95% confidence intervals are included.

	semi_resampling	resampling_before_clustering
	(1)	(2)
ratio_training	−0.023 (−0.252, 0.206)	0.295 (0.142, 0.447)**
	p = 0.843	p = 1.544e-04
ratio_test	0.252 (0.213, 0.29)***	0.346 (0.305, 0.388)***
	p = 5.436e-36	p = 7.190e-58
log10(n_interactions)	−0.011 (−0.032, 0.01)	−0.024 (−0.043, −4.597e-03)*
	p = 0.302	p = 0.015
log10(len_seq)	−0.137 (−0.288, 0.014)	−0.17 (−0.316, −0.025)*
	p = 0.076	p = 0.021
fold1	0.017 (−0.032, 0.066)	−0.123 (−0.17, −0.076)***
	p = 0.497	p = 2.947e-07
fold2	−0.019 (−0.068, 0.031)	0.052 (5.878e-03, 0.099)*
	p = 0.458	p = 0.027
fold3	−0.033 (−0.082, 0.017)	0.038 (−8.623e-03, 0.085)
	p = 0.202	p = 0.11
fold4	−0.01 (−0.06, 0.039)	0.017 (−0.03, 0.064)
	p = 0.68	p = 0.469
fold5	0.035 (−0.014, 0.084)	−0.085 (−0.132, −0.037)**
	p = 0.158	p = 5.370e-04
fold6	−0.055 (−0.105, −6.206e-03)*	−0.025 (−0.073, 0.022)
	p = 0.027	p = 0.295
fold7	−0.171 (−0.219, −0.122)***	1.423e-04 (−0.046, 0.047)
	p = 8.958e-12	p = 0.995
fold8	−0.2 (−0.249, −0.151)***	−0.038 (−0.085, 9.298e-03)
	p = 2.301e-15	p = 0.116
fold9	−0.148 (−0.197, −0.099)***	0.038 (−9.078e-03, 0.084)
	p = 4.477e-09	p = 0.115
Constant	0.907 (0.495, 1.318)**	0.724 (0.338, 1.11)**
	p = 1.665e-05	p = 2.446e-04
Observations	2620	2741
R^2	0.123	0.12
Adjusted R^2	0.118	0.116
Residual Std. Error	0.287 (df = 2606)	0.28 (df = 2727)
F Statistic	28.011*** (df = 13; 2606)	28.722*** (df = 13; 2727)

Note: *p<0.05; **p<1.000e-03; ***p<1e-06

Table S50: Linear models to describe each adjusted performance metric. Standard deviations in parentheses.

	acc	auroc	f1	balanced_acc	mcc
	(1)	(2)	(3)	(4)	(5)
strategyresampling_before_clustering	0.111***	0.114***	0.063***	0.148***	0.202***
	(6.797e-03)	(8.306e-03)	(6.172e-03)	(6.726e-03)	(8.543e-03)
	p = 4.459e-59	p = 1.574e-42	p = 1.135e-24	p = 0.000e+00	p = 0.000e+00
strategysemi_resampling	0.053***	0.011	0.015*	0.068***	0.061***
	(6.905e-03)	(8.479e-03)	(6.270e-03)	(6.833e-03)	(8.678e-03)
	p = 1.885e-14	p = 0.188	p = 0.019	p = 5.502e-23	p = 1.966e-12
strategyresampling_after_clustering	0.057***	0.098***	0.042***	0.12***	0.235***
	(9.236e-03)	(0.01)	(8.387e-03)	(9.140e-03)	(0.012)
	p = 6.266e-10	p = 2.754e-21	p = 6.864e-07	p = 3.564e-39	p = 1.186e-89
log10(n_interactions)	0.04***	0.038***	0.014**	0.014*	0.052***
	(4.375e-03)	(6.018e-03)	(3.972e-03)	(4.329e-03)	(5.498e-03)
	p = 5.057e-20	p = 1.732e-10	p = 3.523e-04	p = 1.485e-03	p = 2.617e-21
log10(len_seq)	−0.091*	−0.034	−1.940e-03	−0.093*	−0.027
	(0.035)	(0.042)	(0.031)	(0.034)	(0.043)
	p = 8.579e-03	p = 0.419	p = 0.951	p = 6.524e-03	p = 0.541
fold1	0.024*	2.277e-03	−0.011	0.014	0.012
	(0.012)	(0.014)	(0.011)	(0.011)	(0.015)
	p = 0.036	p = 0.869	p = 0.297	p = 0.215	p = 0.43
fold2	7.466e-03	0.016	9.307e-03	5.819e-03	0.012
	(0.012)	(0.014)	(0.011)	(0.011)	(0.015)
	p = 0.52	p = 0.26	p = 0.377	p = 0.612	p = 0.401
fold3	0.012	−6.461e-03	−6.538e-03	8.988e-03	−3.534e-03
	(0.012)	(0.014)	(0.011)	(0.012)	(0.015)
	p = 0.322	p = 0.648	p = 0.541	p = 0.441	p = 0.811
fold4	0.015	−8.950e-03	8.932e-03	0.014	3.821e-03
	(0.012)	(0.014)	(0.011)	(0.012)	(0.015)
	p = 0.2	p = 0.53	p = 0.404	p = 0.244	p = 0.797
fold5	6.087e-03	−0.012	−0.018	4.073e-03	−0.012
	(0.012)	(0.014)	(0.011)	(0.012)	(0.015)
	p = 0.604	p = 0.398	p = 0.087	p = 0.726	p = 0.426
fold6	0.017	3.990e-04	−0.017	0.014	0.011
	(0.012)	(0.014)	(0.011)	(0.012)	(0.015)
	p = 0.15	p = 0.977	p = 0.108	p = 0.239	p = 0.474
fold7	9.271e-04	3.367e-03	−0.026*	8.568e-03	0.01
	(0.012)	(0.014)	(0.011)	(0.012)	(0.015)
	p = 0.936	p = 0.809	p = 0.015	p = 0.456	p = 0.484
fold8	0.016	0.011	−0.029*	0.017	0.018
	(0.012)	(0.014)	(0.011)	(0.012)	(0.015)
	p = 0.176	p = 0.424	p = 6.887e-03	p = 0.131	p = 0.213
fold9	7.169e-03	0.011	−0.016	9.026e-03	0.011
	(0.012)	(0.014)	(0.011)	(0.012)	(0.015)
	p = 0.538	p = 0.435	p = 0.131	p = 0.433	p = 0.47
Constant	0.184*	0.077	0.049	0.205*	−0.041
	(0.09)	(0.111)	(0.082)	(0.089)	(0.114)
	p = 0.041	p = 0.486	p = 0.552	p = 0.022	p = 0.719
Observations	9902	7958	9902	9902	9902
R^2	0.042	0.04	0.017	0.056	0.096
Adjusted R^2	0.04	0.039	0.016	0.055	0.095

Note: *p<0.05; **p<1.000e-03; ***p<1e-06

portion of actives per protein in the training and test sets during the model fitting and the predicted proportion of actives. The model performance per protein was also described, pinpointing the most influential factors and characterising the proteins with the most extreme performances.

Four strategies (*no resampling*, *resampling before clustering*, *semi resampling*, *resampling after clustering*) were considered. For each of those, 10 folds of repeated holdout were run, and 5 performance metrics were computed: acc, auroc, f1, balanced_acc, mcc. This led to a total of 10772 values of performance. Since some strategies involved the upsampling method SMOTE, proteins whose sample sizes did not allow upsampling were excluded (table S51).

Table S51: Number of proteins for which performance metrics were computed. The resampling after clustering was the most stringent strategy regarding eligible proteins, since the resampling was carried out after the clustering, which introduced more imbalance.

Strategy	Fold 0	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Fold 6	Fold 7	Fold 8	Fold 9
no_resampling	347	356	359	320	332	331	342	336	341	350
resampling_before_clustering	274	275	284	277	274	256	264	286	271	280
semi_resampling	347	356	359	320	332	331	342	336	341	350
resampling_after_clustering	135	120	125	123	102	116	118	120	122	122

C.2.2 Description of data balance

The data balancing strategy had an impact on the actual data balance, defined as the proportion of active molecules for a protein. Furthermore, specific trends were observed in the original data in the training and test sets, as well as in the values predicted by the deep learning model.

Distributions of the actives ratio

The histograms in figure S82 revealed trends:

- `no resampling` **keeps similar data imbalance in training and test.**
- `resampling before clustering` and `semi resampling` **lead to a more balanced training set, but not so much for the test set.**
- `resampling after clustering` **kept balanced proteins** in both training and test sets.

In addition, **test sets with imbalance tended to magnify** it and create extreme cases (all actives or all inactives), probably due to the combination of the clustering and the lower sample sizes in the test sets compared to training.

Comparing training and test imbalance

Figure S83 revealed both positive, negative and null trends between the training and test set protein balances.

- `no resampling` showed a **positive relation** between both, i.e. proteins were prone to keep their (im)balance in train and test.

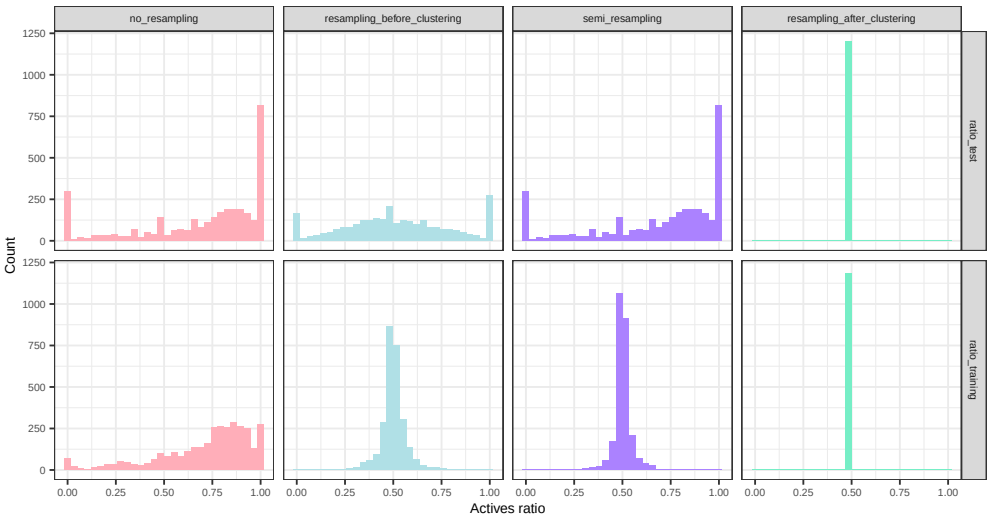

Figure S82: Distribution of the actives ratio in the training set and in the test set (both original and predicted by the deep learning model).

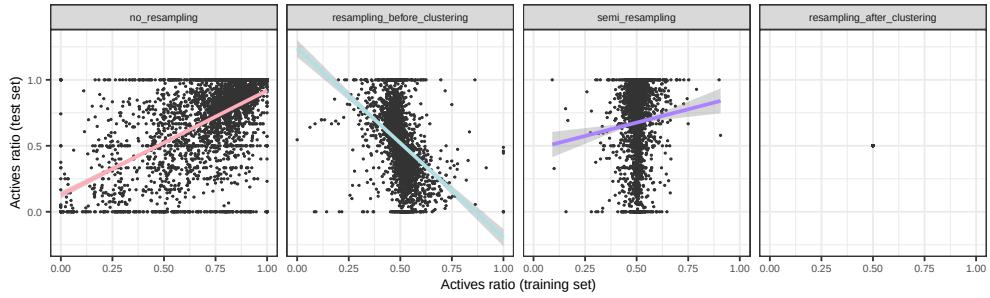

Figure S83: Comparison of the training and test ratios, by resampling strategy. A linear fit line was added per strategy.

- `resampling before clustering` showed an **inverse relationship** instead. This was expected since this strategy started from globally balanced proteins, and after the clustering, an imbalance in one direction in the training set entailed an inverse imbalance in the test set.
- `semi resampling` led to **a slight positive correlation, but weaker than no resampling**.
- `resampling after clustering` always **kept balanced proteins**, by design.

Table S52 displays the Pearson correlation estimate, 95% confidence interval and p-value for each strategy (except `resampling after clustering`, where ratios are constant), further confirming the claims above.

Table S52: Correlations between train and test active ratios. 95% confidence intervals and p-values are shown.

strategy	cor	ci_lower	ci_upper	alternative	pvalue
no_resampling	0.594	0.572	0.616	two.sided	6.73e-320
resampling_before_clustering	-0.386	-0.417	-0.354	two.sided	4.80e-98
semi_resampling	0.067	0.029	0.105	two.sided	5.91e-04
resampling_after_clustering	NA	NA	NA	two.sided	NA

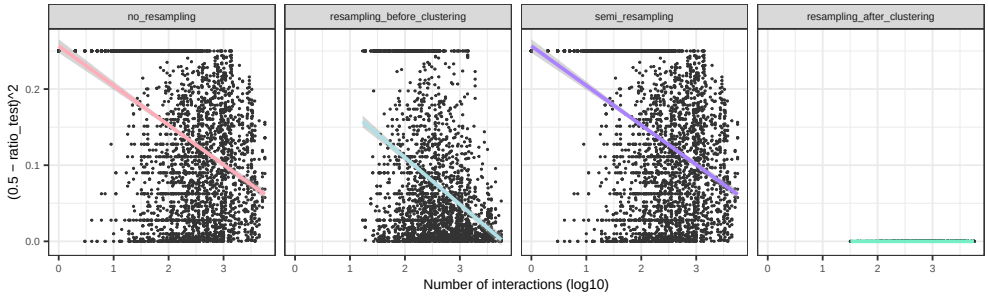

Figure S84: Data imbalance in the test set as a function of the number of available interactions for each protein.

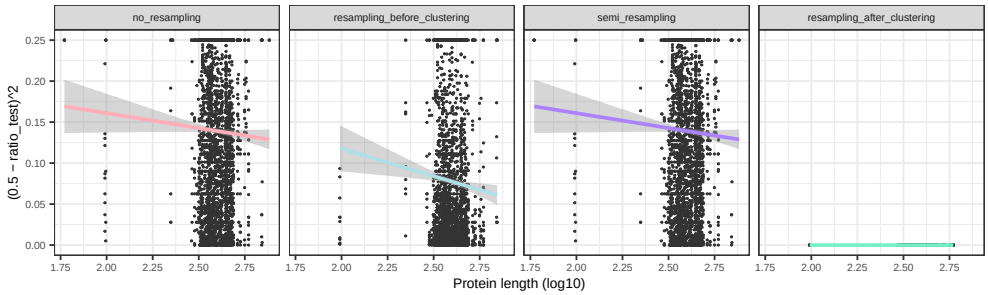

Figure S85: Data imbalance in the test set as a function of the protein length, in amino acids.

Other covariates

The effect of the number of interactions of each protein in its corresponding set and fold (figure S84) and the protein length in amino acids (figure S85) on the test set imbalance was investigated:

- **Proteins with greatest imbalance** (i.e. where `(0.5 - ratio_test)^2` was greatest) **tended to be among those with the least interactions. Linear correlations were significant** (table S53).
- **The sequence length had no obvious effect on the protein imbalance. Linear correlations were not significant** (`no resampling`, `semi resampling`) or significant but low (`resampling before clustering`).

Table S53: Correlations between imbalance (as defined above) and number of interactions. 95% confidence intervals and p-values are shown.

strategy	cor	ci_lower	ci_upper	alternative	pvalue
no_resampling	-0.246	-0.277	-0.214	two.sided	3.72e-48
resampling_before_clustering	-0.305	-0.338	-0.270	two.sided	5.43e-60
semi_resampling	-0.246	-0.277	-0.214	two.sided	3.72e-48
resampling_after_clustering	NA	NA	NA	two.sided	NA

C.2.3 Linear models on predicted proportions

The next key question was to narrow down the factor driving the predicted proportion of actives. The main options under consideration were:

Table S54: Correlations between imbalance (as defined above) and sequence length. 95% confidence intervals and p-values are shown.

strategy	cor	ci_lower	ci_upper	alternative	pvalue
no_resampling	-0.028	-0.062	0.005	two.sided	9.83e-02
resampling_before_clustering	-0.059	-0.097	-0.022	two.sided	1.90e-03
semi_resampling	-0.028	-0.062	0.005	two.sided	9.83e-02
resampling_after_clustering	NA	NA	NA	two.sided	NA

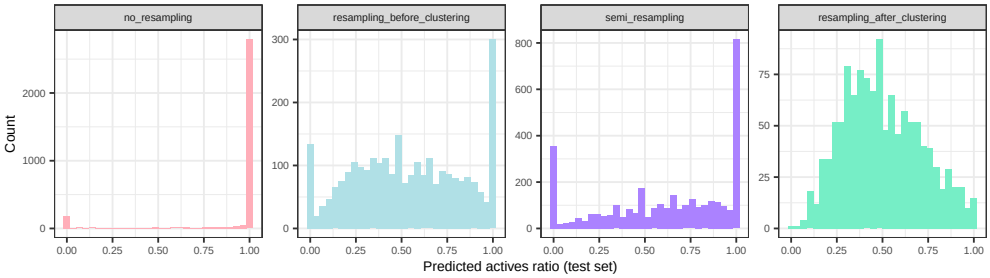

Figure S86: Ratios of the prediction values, after binarization.

1. A constant, global imbalance that the model would learn from the whole dataset.
2. The protein-wise imbalance that the model would learn in the training set.
3. A test set-driven imbalance, based on its actual imbalance.

Distributions of the predicted ratios

After the model predictions in the test set were binarized (actives were those whose probabilities exceeded 0.5), the ratio of predicted actives was computed by protein. This ratio, shown in figure S86, suggested that:

- `no resampling` was noticeably **biased to predict everything as positives**.
- `resampling before clustering` and `semi resampling` **alleviated the imbalance in the predictions, but still retained a spike of proteins where all the compounds were predicted as positives**.
- `resampling after clustering` **kept a wide and symmetric distribution of predicted actives**.

Now, representing together (1) the original training and test ratios, and (2) the predicted ratios in test (figure S87) eased a general qualitative assessment: **the distribution was most resemblant to that of the test proportions to that of the training ones** (except `resampling after clustering`, since those proportions are constant). Table S55 displays how `no resampling` was **highly inclined to predict all positives**, `resampling before clustering` and `semi resampling` **alleviated this phenomenon**, and `resampling after clustering` was **essentially balanced**.

Predicted ratios against training ratios

Figure S88 puts the predicted ratios in context of the training ratios, elucidating a variety of trends:

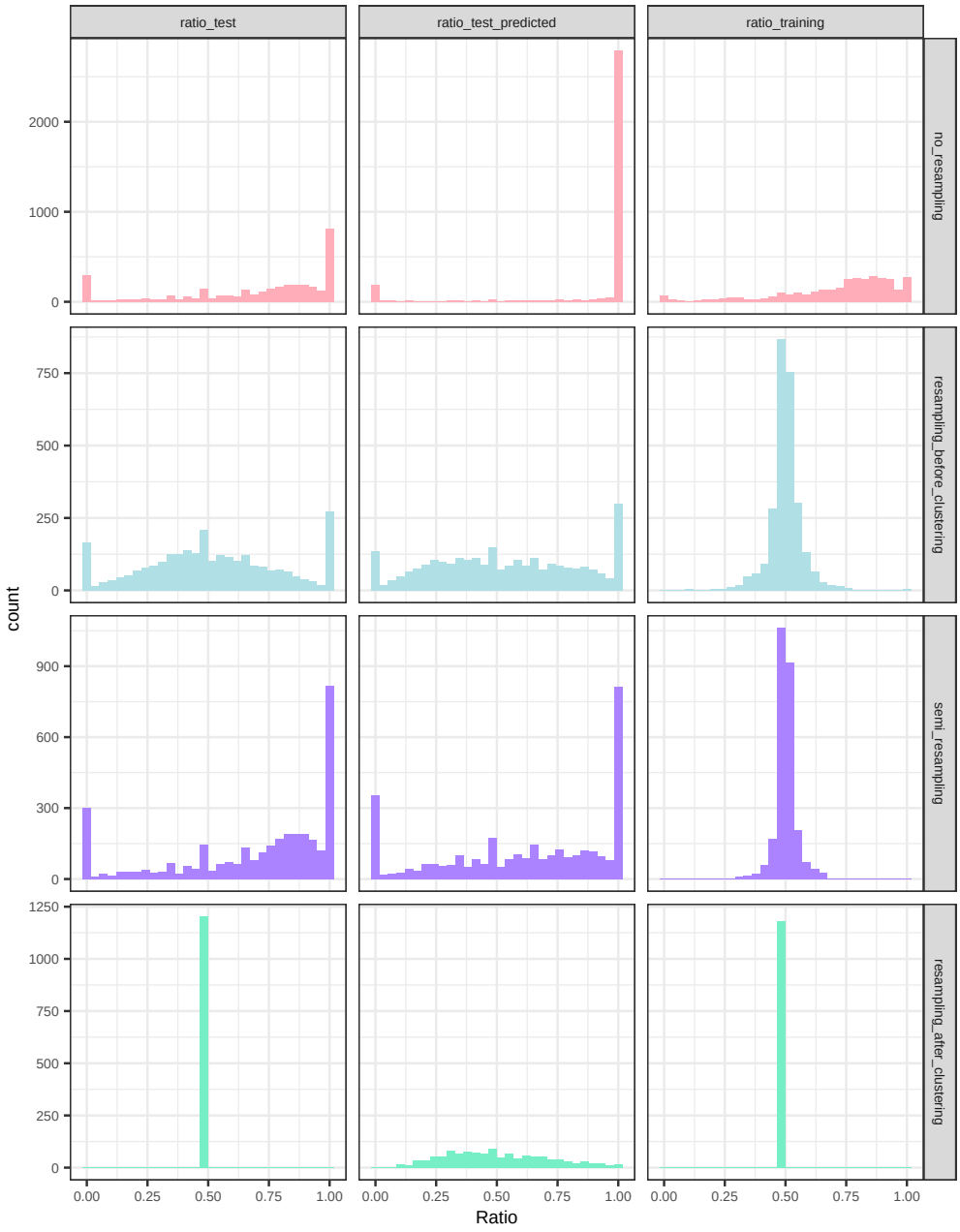

Figure S87: Distribution of the actives ratio in the training set and in the test set (both original and predicted by the deep learning model).

Table S55: Percentage of extreme cases, i.e. proteins with all actives or inactives.

strategy	RatioSet	all_inactives	all_actives	all_extremes
no_resampling	ratio_test	8.8	23.3	32.0
no_resampling	ratio_test_predicted	5.3	80.7	86.0
no_resampling	ratio_training	1.8	7.6	9.4
resampling_before_clustering	ratio_test	5.8	9.7	15.6
resampling_before_clustering	ratio_test_predicted	4.7	10.7	15.4
resampling_before_clustering	ratio_training	0.0	0.2	0.3
semi_resampling	ratio_test	8.8	23.3	32.0
semi_resampling	ratio_test_predicted	10.4	23.3	33.7
semi_resampling	ratio_training	0.0	0.0	0.0
resampling_after_clustering	ratio_test	0.0	0.0	0.0
resampling_after_clustering	ratio_test_predicted	0.1	1.1	1.2
resampling_after_clustering	ratio_training	0.0	0.0	0.0

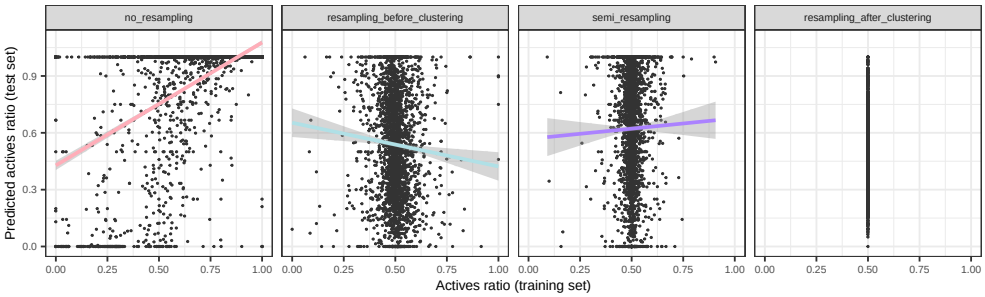

Figure S88: Predicted ratios, as a function of training ratios.

- `no resampling`: **positive trend between the training and the predicted ratio**, but since the training and the test ratio also positively correlated (figure 35), the latter could be the one driving the predicted ratio of positives.
- `resampling after clustering` had a **constant training ratio**, meaning that the predicted ratio was not explainable by differences in training ratios.
- `resampling before clustering` showed instead a **negative relation between the training and the predicted ratio**. But since the former and the test ratio also anticorrelated (figure 35, the simplest explanation was that the test ratio drove the predicted test ratio.
- `semi resampling` showed **no apparent correlation** between the predicted ratio and the training ratio.

The significance of the linear correlation backs up all the claims above (table S56).

Table S56: Correlations between train and predicted test active ratios. 95% confidence intervals and p-values are shown.

strategy	cor	ci_lower	ci_upper	alternative	pvalue
no_resampling	0.561	0.538	0.584	two.sided	4.11e-278
resampling_before_clustering	-0.058	-0.095	-0.021	two.sided	2.37e-03
semi_resampling	0.017	-0.021	0.056	two.sided	3.76e-01
resampling_after_clustering	NA	NA	NA	two.sided	NA

Linear models

The predicted ratio of actives r_{pred} was modelled through the following quasibinomial generalized linear models, stratified by strategy:

$$r_{pred} \sim r_{training} + r_{test} + \log10(n_{int}) + \log10(n_{seq}) + k_{fold}$$

The main variables of interest are the actual ratios in the training $r_{training}$ and in test r_{test}, both numeric between 0 and 1. As additional covariates, the number of interactions n_{int} and the sequence length n_{seq} (numerical) and the fold number k_{fold} (categorical) were also included.

IN SEMI RESAMPLING OR RESAMPLING BEFORE CLUSTERING **Due to the only slight correlation between training and test ratios** (figure 35), the `semi resampling` strategy is the **ideal scenario to disentangle their effects** on the predicted ratio of actives (see model in table S57). This additive model suggests:

- The **test ratio is driving the predicted proportions, rather than the training ratio**.
- `n_interactions`: **the term is not significant in GPCRs**.

Table S57 also shows the additive model for `resampling before clustering`. This strategy showed negative correlation between training and test ratios, also providing a reasonably good scenario to distinguish their effects.

- **This model confirms both conclusions from the model** in the `semi resampling` strategy. **The ratio in training is also significant for GPCRs, but lower significance and estimate than the ratio in test; number of interactions slightly significant now.**

IN NO RESAMPLING The explanatory linear model under the `no resampling` strategy (table S58) **suffers from the positive correlation between training and test ratios, which can be confounded**.

- Both `training_ratio` and `test_ratio` show a **positive effect** on the predicted fraction of actives.
- Although the **estimate is larger and more significant** for `training_ratio`, the **confounding effect and the very skewed distribution of the predicted ratios deems this model inconclusive**.

Conclusions

- Data imbalance exists in all strategies but in `resampling after clustering`, where balance is enforced.
- The correlation between a protein's ratio in train and test is positive in `no resampling`, negative in `resampling before clustering` and null in `semi resampling` and `resampling after clustering`.
- The main factor driving the ratio of actives in the model predictions, per protein, is the actual ratio of positives in the test set. Their distributions are resemblant, and linear models confirm the association.

All of them apply to GPCRs.

Table S57: Linear models to describe the predicted active ratio for the *semi resampling* and the *resampling before clustering* strategies. Significance and 95% confidence intervals are included.

	semi_resampling	resampling_before_clustering
	(1)	(2)
ratio_training	−0.106 (−1.147, 0.935)	1.269 (0.613, 1.926)**
	p = 0.842	p = 1.553e-04
ratio_test	1.093 (0.92, 1.266)***	1.456 (1.274, 1.637)***
	p = 3.309e-34	p = 1.501e-53
log10(n_interactions)	−0.055 (−0.15, 0.04)	−0.102 (−0.183, −0.021)*
	p = 0.259	p = 0.013
log10(len_seq)	−0.625 (−1.315, 0.066)	−0.728 (−1.347, −0.109)*
	p = 0.076	p = 0.021
fold1	0.082 (−0.145, 0.309)	−0.512 (−0.711, −0.313)***
	p = 0.477	p = 4.829e-07
fold2	−0.085 (−0.309, 0.139)	0.219 (0.022, 0.417)*
	p = 0.455	p = 0.03
fold3	−0.145 (−0.371, 0.081)	0.162 (−0.037, 0.362)
	p = 0.209	p = 0.11
fold4	−0.044 (−0.27, 0.183)	0.073 (−0.125, 0.271)
	p = 0.706	p = 0.471
fold5	0.172 (−0.055, 0.4)	−0.354 (−0.557, −0.152)**
	p = 0.138	p = 6.138e-04
fold6	−0.249 (−0.471, −0.027)*	−0.106 (−0.307, 0.095)
	p = 0.028	p = 0.3
fold7	−0.732 (−0.949, −0.515)***	−1.350e-03 (−0.197, 0.195)
	p = 4.372e-11	p = 0.989
fold8	−0.851 (−1.069, −0.633)***	−0.156 (−0.355, 0.042)
	p = 2.873e-14	p = 0.123
fold9	−0.638 (−0.858, −0.419)***	0.157 (−0.041, 0.356)
	p = 1.286e-08	p = 0.12
Constant	1.848 (−0.032, 3.727)	0.961 (−0.685, 2.606)
	p = 0.054	p = 0.253
Observations	2620	2741

Note: *p<0.05; **p<1.000e-03; ***p<1e-06

Table S58: Linear models to describe the predicted active ratio for the *no resampling* strategy. Significance and 95% confidence intervals are included.

	no_resampling
ratio_training	6.695 (5.63, 7.76)***
	p = 3.913e-34
ratio_test	1.336 (0.663, 2.009)**
	p = 1.011e-04
log10(n_interactions)	−1.147 (−1.487, −0.806)***
	p = 4.808e-11
log10(len_seq)	3.459 (1.24, 5.679)*
	p = 2.273e-03
fold1	−1.024 (−1.928, −0.12)*
	p = 0.026
fold2	−0.455 (−1.381, 0.47)
	p = 0.335
fold3	−0.879 (−1.795, 0.037)
	p = 0.06
fold4	−0.221 (−1.209, 0.768)
	p = 0.662
fold5	−1.02 (−1.923, −0.118)*
	p = 0.027
fold6	−1.615 (−2.481, −0.749)**
	p = 2.620e-04
fold7	−0.879 (−1.791, 0.034)
	p = 0.059
fold8	−0.403 (−1.363, 0.557)
	p = 0.411
fold9	−0.469 (−1.392, 0.454)
	p = 0.319
Constant	−7.999 (−13.868, −2.13)*
	p = 7.593e-03
Observations	3359
Note:	*p<0.05; **p<1.000e-03; ***p<1e-06

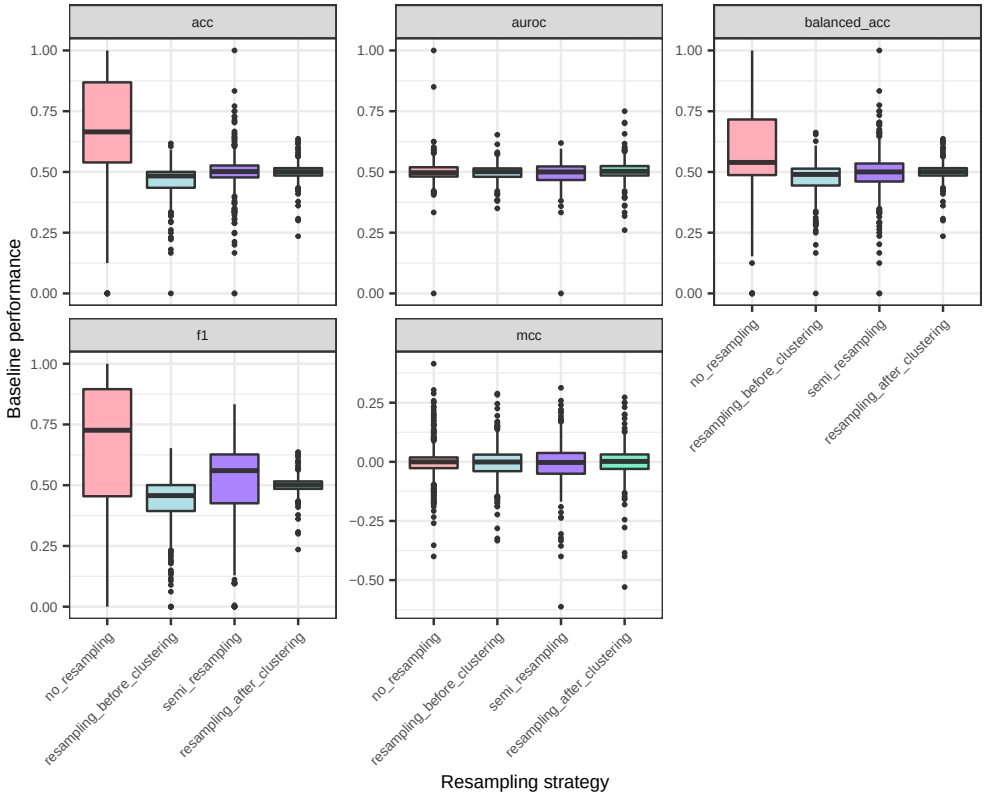

Figure S89: Performance metrics for imbalance-aware random baselines. Data points correspond to proteins, averaged over folds.

C.2.4 Description of baseline performance

Before evaluating the deep learning model, the performance metrics of the baselines were characterised, in order to pinpoint imbalance-sensitive and insensitive metrics. Metrics were called imbalance-sensitive if the imbalance-aware random baseline exhibited different performances between resampling strategies.

Descriptive plot

Figure S89 shows a fold-averaged picture of the metrics by protein. **Visual inspection suggested that accuracy, F1 and possibly balanced accuracy were affected by the data imbalance**. **F1 is the most apparent case**, see the quartiles in table S59.

Table S59: Quartiles for the baseline F1-scores.

strategy	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
no_resampling	0.000	0.454	0.726	0.647	0.895	1.000
resampling_before_clustering	0.000	0.394	0.457	0.430	0.501	0.653
semi_resampling	0.000	0.426	0.560	0.506	0.627	0.833
resampling_after_clustering	0.235	0.485	0.501	0.498	0.516	0.636

Linear models

Formally, each performance metric was described with the following linear model:

$$\text{metric} \sim \text{strategy} + \log10(n_{int}) + \log10(n_{seq}) + k_{fold}$$

The response was the quantitative metric of interest (one model per metric), while `strategy` was categorical with the following possibilities: `no resampling`, `resampling after clustering`, `resampling before clustering`, `semi resampling`. Additional covariates included the number of interactions n_{int} and the sequence length n_{seq} (numerical) and the fold number k_{fold} (categorical). The `strategy` variable was tested with a type 3 ANOVA, being **significant** with $p < 0.05$ for `acc`, `f1` and `balanced_acc` (table S60).

Table S60: ANOVA p-values for including the resampling strategy as a regressor. Significant p-values imply that differences exist between resampling strategies.

strategy	variable	Sum Sq	Df	F value	Pr(>F)
acc	strategy	79.2893867	3	7.37e+02	0.00e+00
auroc	strategy	0.0060204	3	6.74e-02	9.77e-01
balanced_acc	strategy	15.7514288	3	1.28e+02	1.81e-81
f1	strategy	100.8229851	3	5.60e+02	0.00e+00
mcc	strategy	0.0338470	3	2.34e-01	8.73e-01

Based on this, metrics were divided in two types:

- Those where the baseline was different between strategies, i.e. imbalance-sensitive: `acc`, `f1` and `balanced_acc`. Therefore, before comparing strategies, the baseline performance needed to be accounted for.
- Those where the baseline was constant, i.e. imbalance-insensitive: `auroc`, `mcc`. Here we could compare strategies directly.

All applies to GPCRs as well.

C.2.5 Description of deep learning model performance

An overview of fold-averaged performances is displayed in figure S90, where strategies are paired with their baselines. This illustrates the **issue of direct strategy comparison** with imbalance-sensitive metrics, which was especially visible for the F1-score. Some metrics are undefined in edge cases (e.g. AUROC when only actives or only inactives are available); table S61 summarizes the number of proteins, added over folds, whose metrics were computable.

Absolute, baseline-naive performance

Anologous to the baseline performance models, absolute metric models (not accounting for baselines) were fitted:

Table S61: Number of computable performance measures. AUROC was undefined for proteins with all actives or unactives in the test set, hence its lower counts.

strategy	acc	auroc	f1	balanced_acc	mcc
no_resampling	3414	2320	3414	3414	3414
resampling_before_clustering	2741	2314	2741	2741	2741
semi_resampling	3414	2320	3414	3414	3414
resampling_after_clustering	1203	1203	1203	1203	1203

$$\text{metric} \sim \text{strategy} + \log10(n_{\texttt{int}}) + \log10(n_{\texttt{seq}}) + k_{\texttt{fold}}$$

The `strategy` covariate was **always significant** in a type 3 ANOVA (table S62). The models, summarized in S63, showed **different behaviour in imbalance-sensitive and insensitive metrics**. Pairwise comparisons of the strategy coefficients using Tukey's method would point to **two different pictures** (figure S91), further confirmed when prioritizing the strategies according to their expected performance through the linear models (table S64 and figure S92):

- Accuracy and F1-score suggested that `no resampling` was the **best strategy**, but this was confounded by the fact that it also held the **highest baselines**.
- AUROC, MCC and balanced accuracy showed instead that `resampling before clustering` and `resampling after clustering` held the **highest performance estimates**.

Table S62: ANOVA p-values for including the resampling strategy as a regressor in the performance models.

strategy	variable	Sum Sq	Df	F value	Pr(>F)
acc	strategy	26.661749	3	1.26e+02	2.90e-80
auroc	strategy	21.408275	3	1.39e+02	1.40e-87
f1	strategy	61.491300	3	2.10e+02	2.20e-132
balanced_acc	strategy	7.868812	3	4.27e+01	2.08e-27
mcc	strategy	82.398075	3	4.78e+02	1.23e-291

Table S63: Linear models to describe each performance metric. Standard deviations in parentheses.

	acc	auroc	f1	balanced_acc	mcc
	(1)	(2)	(3)	(4)	(5)
strategyresampling_before_clustering	−0.096***	0.114***	−0.182***	0.051***	0.2***
	(6.883e-03)	(6.673e-03)	(8.100e-03)	(6.427e-03)	(6.214e-03)
	p = 4.956e-44	p = 8.545e-65	p = 0.000e+00	p = 1.253e-15	p = 0.000e+00
strategysemi_resampling	−0.112***	0.013*	−0.14***	−0.01	0.057***
	(6.426e-03)	(6.662e-03)	(7.562e-03)	(6.000e-03)	(5.801e-03)
	p = 3.801e-67	p = 0.048	p = 7.533e-76	p = 0.083	p = 1.271e-22
strategyresampling_after_clustering	−0.116***	0.101***	−0.171***	0.054***	0.232***
	(9.213e-03)	(8.193e-03)	(0.011)	(8.602e-03)	(8.317e-03)
	p = 3.303e-36	p = 1.520e-34	p = 1.286e-55	p = 4.246e-10	p = 0.000e+00
log10(n_interactions)	0.029***	0.046***	0.081***	−0.025***	0.054***
	(3.725e-03)	(4.573e-03)	(4.383e-03)	(3.478e-03)	(3.362e-03)
	p = 6.242e-15	p = 2.260e-23	p = 1.546e-75	p = 2.961e-13	p = 3.935e-58
log10(len_seq)	−0.084*	−0.054	−0.019	−0.088*	−0.019
	(0.033)	(0.033)	(0.039)	(0.031)	(0.03)
	p = 0.011	p = 0.108	p = 0.624	p = 4.142e-03	p = 0.519
fold1	0.014	0.012	−7.494e-03	3.971e-03	0.013
	(0.011)	(0.011)	(0.013)	(0.011)	(0.01)
	p = 0.203	p = 0.276	p = 0.573	p = 0.707	p = 0.195
fold2	0.015	0.021	9.361e-03	0.011	0.014
	(0.011)	(0.011)	(0.013)	(0.011)	(0.01)
	p = 0.193	p = 0.05	p = 0.479	p = 0.275	p = 0.176
fold3	0.011	5.610e-03	−5.323e-03	9.151e-03	2.785e-03
	(0.011)	(0.011)	(0.014)	(0.011)	(0.01)
	p = 0.329	p = 0.617	p = 0.694	p = 0.393	p = 0.788
fold4	0.034*	3.539e-03	0.017	0.028*	0.014
	(0.011)	(0.011)	(0.014)	(0.011)	(0.01)
	p = 3.126e-03	p = 0.754	p = 0.206	p = 8.064e-03	p = 0.163
fold5	0.016	−1.805e-03	−0.011	9.616e-03	−3.018e-03
	(0.011)	(0.011)	(0.014)	(0.011)	(0.01)
	p = 0.171	p = 0.871	p = 0.437	p = 0.37	p = 0.771
fold6	0.017	0.014	−0.019	0.014	0.021*
	(0.011)	(0.011)	(0.013)	(0.011)	(0.01)
	p = 0.128	p = 0.199	p = 0.146	p = 0.183	p = 0.044
fold7	2.239e-03	0.011	−0.025	8.648e-03	0.019
	(0.011)	(0.011)	(0.013)	(0.011)	(0.01)
	p = 0.844	p = 0.301	p = 0.057	p = 0.415	p = 0.071
fold8	0.015	0.024*	−0.025	0.015	0.026*
	(0.011)	(0.011)	(0.013)	(0.011)	(0.01)
	p = 0.19	p = 0.03	p = 0.059	p = 0.159	p = 0.012
fold9	−4.569e-04	0.014	−0.033*	5.402e-03	0.01
	(0.011)	(0.011)	(0.013)	(0.011)	(0.01)
	p = 0.968	p = 0.198	p = 0.013	p = 0.609	p = 0.323
Constant	0.868***	0.596***	0.609***	0.868***	−0.072
	(0.085)	(0.087)	(0.101)	(0.08)	(0.077)
	p = 3.968e-24	p = 8.565e-12	p = 1.434e-09	p = 2.065e-27	p = 0.352
Observations	10772	8157	10772	10772	10772
R^2	0.039	0.067	0.076	0.015	0.169
Adjusted R^2	0.037	0.065	0.075	0.014	0.168

Note: *p<0.05; **p<1.000e-03; ***p<1e-06

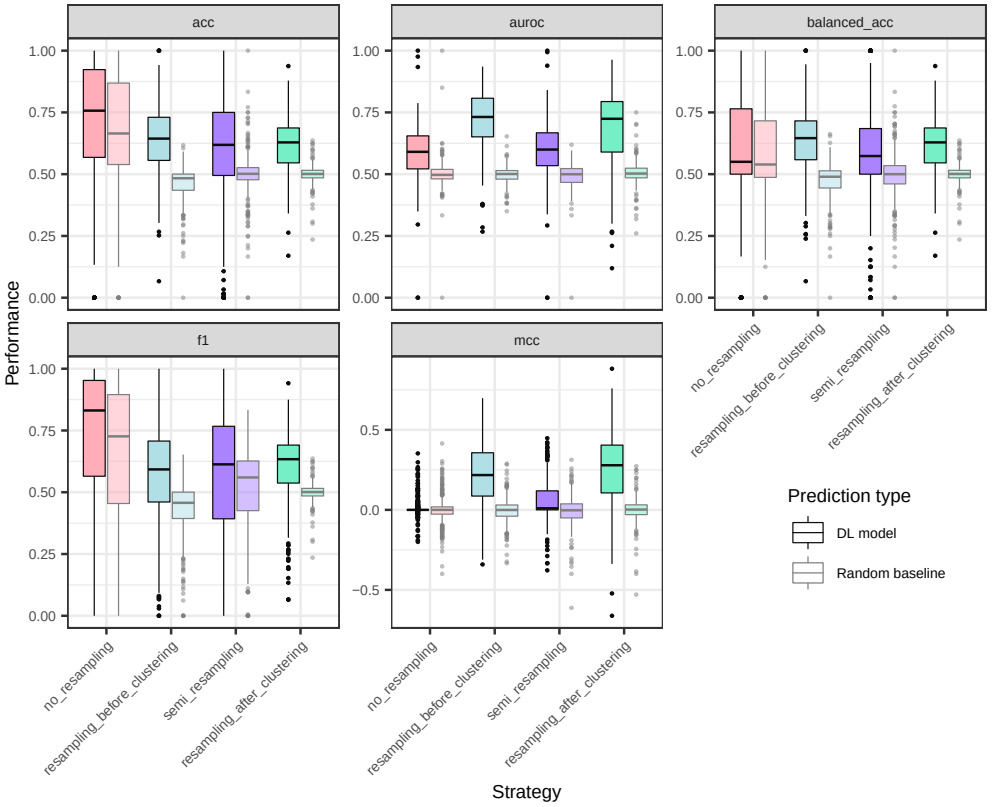

Figure S90: Performance metrics for balancing strategies and their corresponding imbalance-aware random baselines. Data points correspond to proteins, averaged over folds.

Table S64: Expected absolute performances, by metric and strategy, with 95% confidence intervals.

metric	strategy	emmean	SE	lower.CL	upper.CL
acc	no_resampling	0.746	5.030e-03	0.736	0.755
acc	resampling_before_clustering	0.649	5.202e-03	0.639	0.659
acc	semi_resampling	0.633	5.030e-03	0.624	0.643
acc	resampling_after_clustering	0.629	7.660e-03	0.614	0.644
auroc	no_resampling	0.597	4.861e-03	0.588	0.607
auroc	resampling_before_clustering	0.711	4.831e-03	0.702	0.721
auroc	semi_resampling	0.610	4.861e-03	0.601	0.620
auroc	resampling_after_clustering	0.698	6.550e-03	0.685	0.711
f1	no_resampling	0.780	5.919e-03	0.769	0.792
f1	resampling_before_clustering	0.598	6.121e-03	0.586	0.610
f1	semi_resampling	0.640	5.919e-03	0.628	0.652
f1	resampling_after_clustering	0.609	9.013e-03	0.591	0.627
balanced_acc	no_resampling	0.579	4.697e-03	0.569	0.588
balanced_acc	resampling_before_clustering	0.630	4.857e-03	0.621	0.640
balanced_acc	semi_resampling	0.568	4.697e-03	0.559	0.577
balanced_acc	resampling_after_clustering	0.632	7.152e-03	0.618	0.646
mcc	no_resampling	0.044	4.541e-03	0.035	0.053
mcc	resampling_before_clustering	0.244	4.696e-03	0.235	0.253
mcc	semi_resampling	0.101	4.541e-03	0.092	0.110
mcc	resampling_after_clustering	0.277	6.915e-03	0.263	0.290

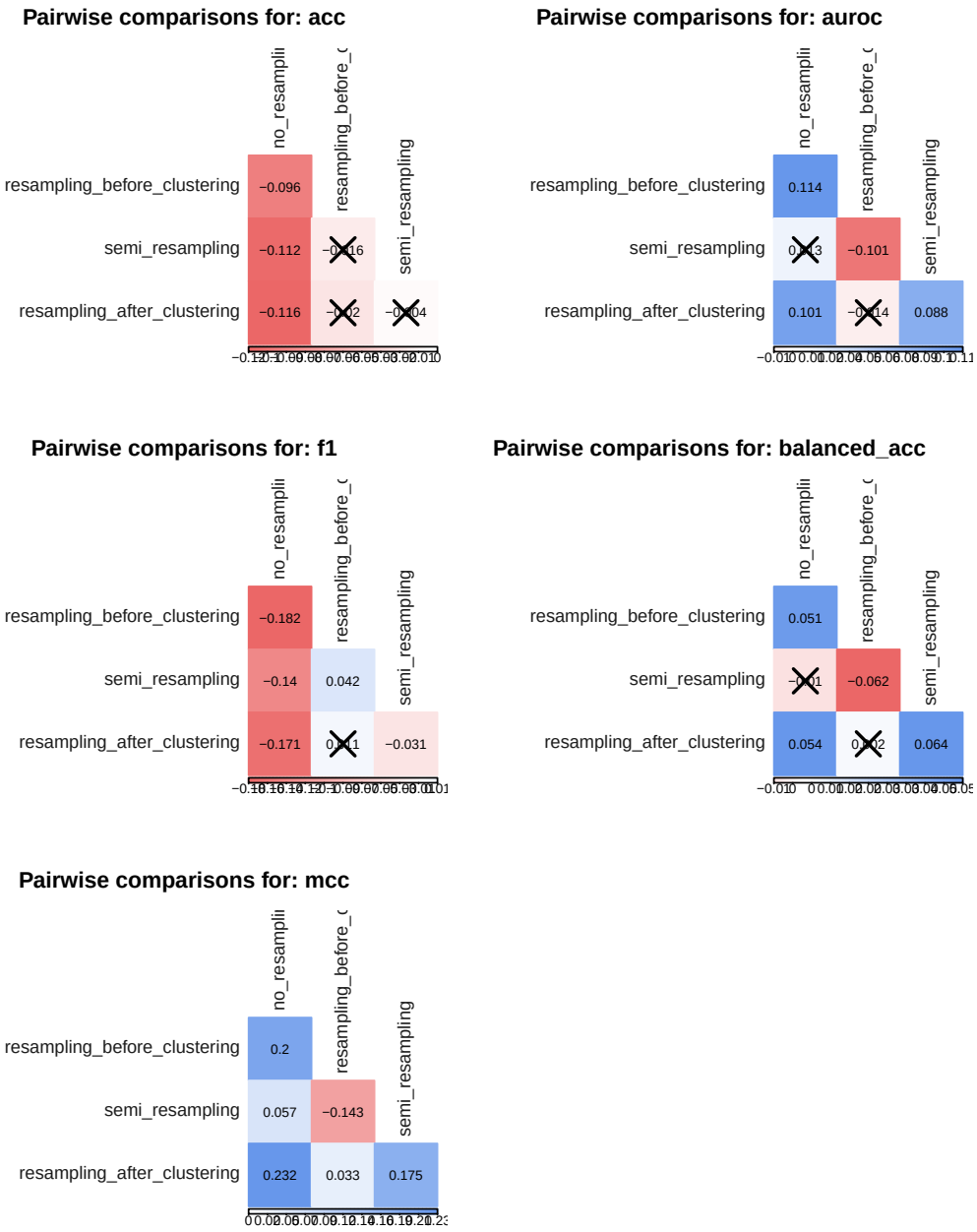

Figure S91: Pairwise comparison of strategy performance using Tukey method.

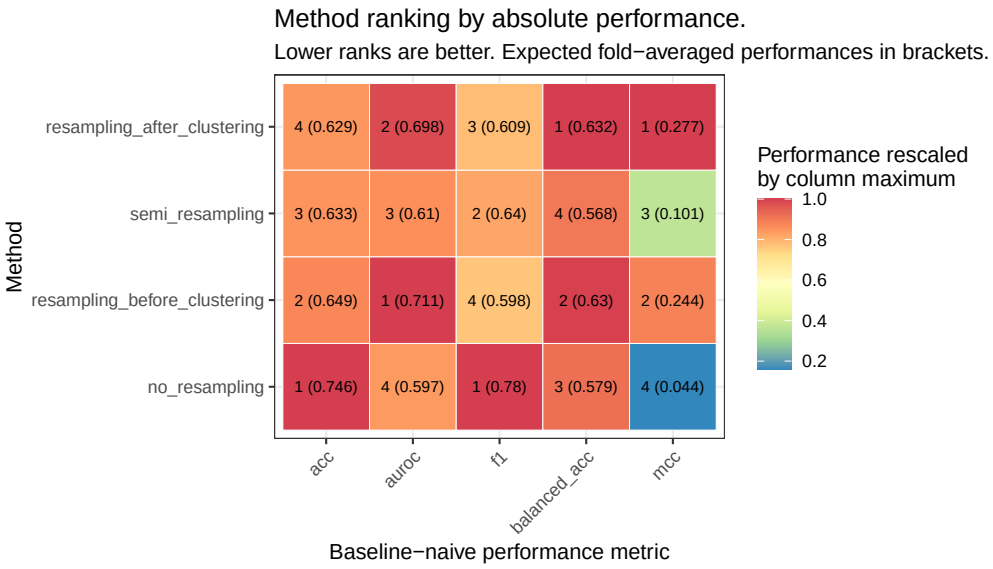

Figure S92: Method ranking according to the linear model predicted performances for each metric. Baseline metrics were ignored.

Baseline-adjusted performance

To address the pitfalls of the direct comparison of metrics whose baselines may differ, baseline-adjusted performance metrics were defined and modelled analogously. Specifically:

$$\text{adj\_metric} = \text{metric} - \text{baseline}$$

A descriptive plot of the the adjusted metrics (figure S93) pointed to a **scenario diffent than that of unadjusted ones** (figure S90).

Adjusted performance metrics were described with the following linear model:

$$\text{adj\_metric} \sim \text{strategy} + \log10(n_{int}) + \log10(n_{seq}) + k_{fold}$$

Note that while all metrics but `mcc` were non-negative, the adjusted metrics could show negative values when the performance of the DL model was lower than that of the baseline.

Again, `strategy` covariate was **always significant** in a type 3 ANOVA (table S65). **Baseline adjustment brought a uniform behaviour across the models** (table S66), further confirmed in pairwise coefficient comparison (Tukey's method, figure S94) and in their expected performance (table S67 and figure S95):

- `resampling before clustering` and `resampling after clustering` had the **highest performance estimates, followed by** `semi resampling` and finally by `no resampling`.
- This **picture was analogous to that of the non-adjusted performance metrics that were imbalance-insensitive** (AUROC, MCC).

Conclusions drawn from the baseline-adjusted performance analysis:

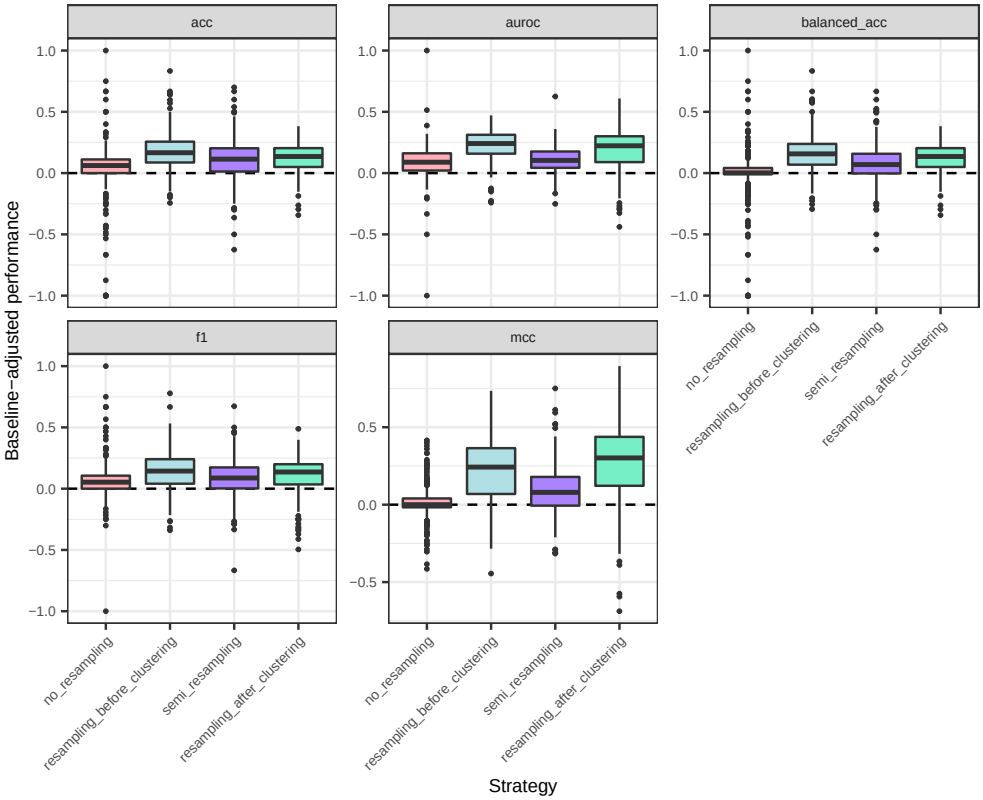

Figure S93: Baseline-adjusted performance metrics for balancing strategies. Data points correspond to proteins, averaged over folds.

Table S65: ANOVA p-values for including the resampling strategy as a regressor in the adjusted performance models.

strategy	variable	Sum Sq	Df	F value	Pr(>F)
acc	strategy	18.137218	3	8.88e+01	1.12e-56
auroc	strategy	21.118128	3	8.83e+01	3.59e-56
f1	strategy	6.556575	3	3.89e+01	5.40e-25
balanced_acc	strategy	34.369289	3	1.72e+02	1.50e-108
mcc	strategy	83.131344	3	2.58e+02	6.13e-161

- The largest impact in performance estimates was the application of **data augmentation to the test set**: `resampling_before_clustering` and `resampling_after_clustering` tended to outperform `semi_resampling` and `no_resampling` (Tukey's method, $p < 0.05$, figure S94). However, augmenting the test set might not faithfully reflect new data anymore, and could **artificially inflate the peformance estimates**.
- `semi resampling` outperformed `no resampling` in **three out of five, being the other two non-significant** (Tukey's method, $p < 0.05$, figure S94), which **supports data augmentation usefulness** even if the data balance in the test set differs from that of the training set. This was **consistent with the observation that the predicted proportion of positives of the PCM model was mainly driven by the actual data balance in the test set, rather than that of the training set. Combined with the**

Table S66: Linear models to describe each adjusted performance metric. Standard deviations in parentheses.

	acc	auroc	f1	balanced_acc	mcc
	(1)	(2)	(3)	(4)	(5)
strategyresampling_before_clustering	0.111***	0.114***	0.063***	0.148***	0.202***
	(6.797e-03)	(8.306e-03)	(6.172e-03)	(6.726e-03)	(8.543e-03)
	p = 4.459e-59	p = 1.574e-42	p = 1.135e-24	p = 0.000e+00	p = 0.000e+00
strategysemi_resampling	0.053***	0.011	0.015*	0.068***	0.061***
	(6.905e-03)	(8.479e-03)	(6.270e-03)	(6.833e-03)	(8.678e-03)
	p = 1.885e-14	p = 0.188	p = 0.019	p = 5.502e-23	p = 1.966e-12
strategyresampling_after_clustering	0.057***	0.098***	0.042***	0.12***	0.235***
	(9.236e-03)	(0.01)	(8.387e-03)	(9.140e-03)	(0.012)
	p = 6.266e-10	p = 2.754e-21	p = 6.864e-07	p = 3.564e-39	p = 1.186e-89
log10(n_interactions)	0.04***	0.038***	0.014**	0.014*	0.052***
	(4.375e-03)	(6.018e-03)	(3.972e-03)	(4.329e-03)	(5.498e-03)
	p = 5.057e-20	p = 1.732e-10	p = 3.523e-04	p = 1.485e-03	p = 2.617e-21
log10(len_seq)	−0.091*	−0.034	−1.940e-03	−0.093*	−0.027
	(0.035)	(0.042)	(0.031)	(0.034)	(0.043)
	p = 8.579e-03	p = 0.419	p = 0.951	p = 6.524e-03	p = 0.541
fold1	0.024*	2.277e-03	−0.011	0.014	0.012
	(0.012)	(0.014)	(0.011)	(0.011)	(0.015)
	p = 0.036	p = 0.869	p = 0.297	p = 0.215	p = 0.43
fold2	7.466e-03	0.016	9.307e-03	5.819e-03	0.012
	(0.012)	(0.014)	(0.011)	(0.011)	(0.015)
	p = 0.52	p = 0.26	p = 0.377	p = 0.612	p = 0.401
fold3	0.012	−6.461e-03	−6.538e-03	8.988e-03	−3.534e-03
	(0.012)	(0.014)	(0.011)	(0.012)	(0.015)
	p = 0.322	p = 0.648	p = 0.541	p = 0.441	p = 0.811
fold4	0.015	−8.950e-03	8.932e-03	0.014	3.821e-03
	(0.012)	(0.014)	(0.011)	(0.012)	(0.015)
	p = 0.2	p = 0.53	p = 0.404	p = 0.244	p = 0.797
fold5	6.087e-03	−0.012	−0.018	4.073e-03	−0.012
	(0.012)	(0.014)	(0.011)	(0.012)	(0.015)
	p = 0.604	p = 0.398	p = 0.087	p = 0.726	p = 0.426
fold6	0.017	3.990e-04	−0.017	0.014	0.011
	(0.012)	(0.014)	(0.011)	(0.012)	(0.015)
	p = 0.15	p = 0.977	p = 0.108	p = 0.239	p = 0.474
fold7	9.271e-04	3.367e-03	−0.026*	8.568e-03	0.01
	(0.012)	(0.014)	(0.011)	(0.012)	(0.015)
	p = 0.936	p = 0.809	p = 0.015	p = 0.456	p = 0.484
fold8	0.016	0.011	−0.029*	0.017	0.018
	(0.012)	(0.014)	(0.011)	(0.012)	(0.015)
	p = 0.176	p = 0.424	p = 6.887e-03	p = 0.131	p = 0.213
fold9	7.169e-03	0.011	−0.016	9.026e-03	0.011
	(0.012)	(0.014)	(0.011)	(0.012)	(0.015)
	p = 0.538	p = 0.435	p = 0.131	p = 0.433	p = 0.47
Constant	0.184*	0.077	0.049	0.205*	−0.041
	(0.09)	(0.111)	(0.082)	(0.089)	(0.114)
	p = 0.041	p = 0.486	p = 0.552	p = 0.022	p = 0.719
Observations	9902	7958	9902	9902	9902
R^2	0.042	0.04	0.017	0.056	0.096
Adjusted R^2	0.04	0.039	0.016	0.055	0.095
Note:					*p<0.05; **p<1.000e-03; ***p<1e-06

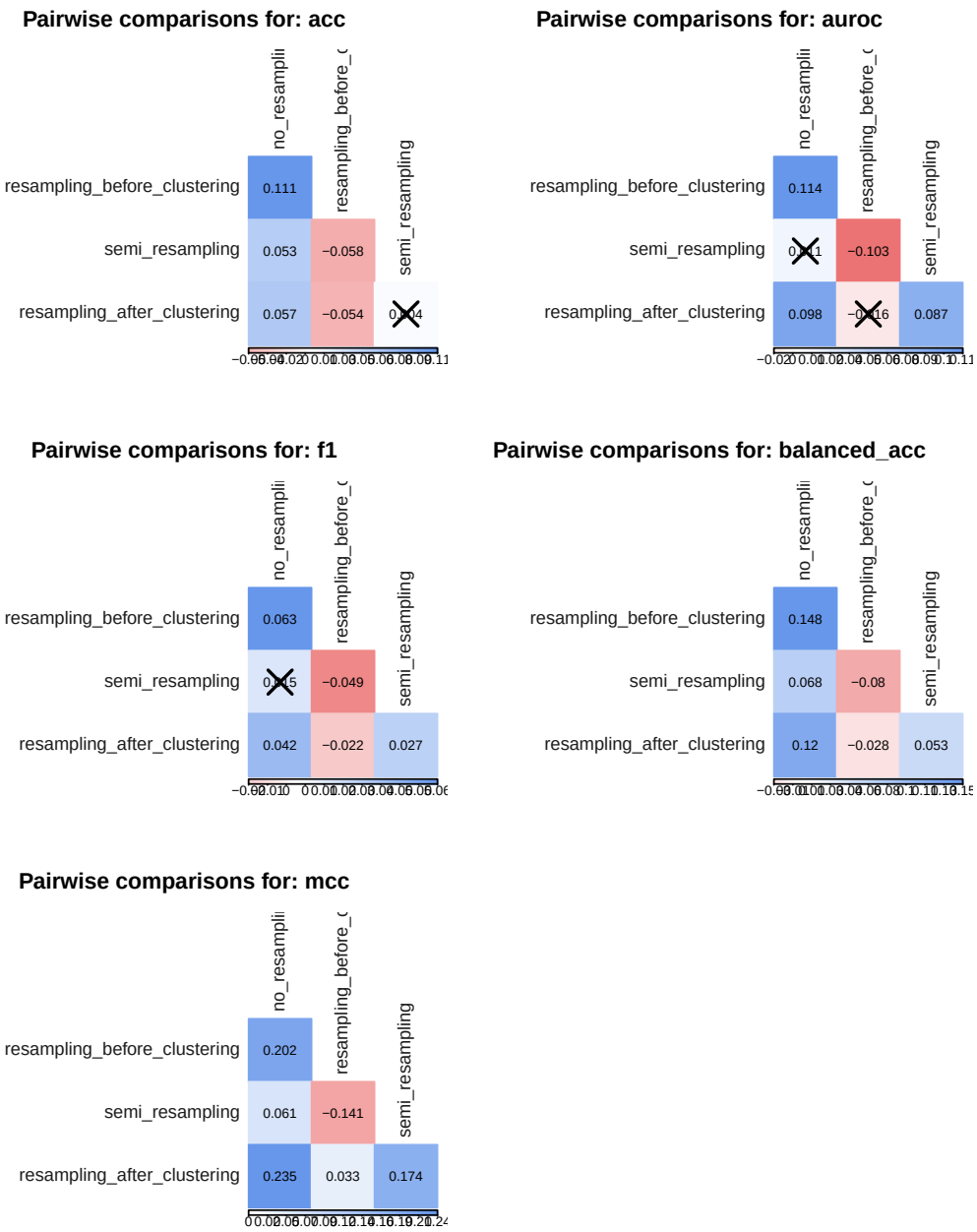

Figure S94: Pairwise comparison of strategy adjusted performance using Tukey method.

Table S67: Expected adjusted performances, by metric and strategy, with 95% confidence intervals.

metric	strategy	emmean	SE	lower.CL	upper.CL
acc	no_resampling	0.074	5.172e-03	0.064	0.084
acc	resampling_before_clustering	0.185	5.210e-03	0.175	0.195
acc	semi_resampling	0.127	5.278e-03	0.117	0.137
acc	resampling_after_clustering	0.131	7.595e-03	0.116	0.146
auroc	no_resampling	0.101	6.130e-03	0.089	0.113
auroc	resampling_before_clustering	0.215	6.079e-03	0.203	0.227
auroc	semi_resampling	0.112	6.232e-03	0.100	0.124
auroc	resampling_after_clustering	0.198	8.222e-03	0.182	0.214
f1	no_resampling	0.074	4.696e-03	0.065	0.083
f1	resampling_before_clustering	0.138	4.731e-03	0.128	0.147
f1	semi_resampling	0.089	4.792e-03	0.079	0.098
f1	resampling_after_clustering	0.116	6.897e-03	0.102	0.129
balanced_acc	no_resampling	0.012	5.118e-03	0.002	0.022
balanced_acc	resampling_before_clustering	0.160	5.156e-03	0.150	0.170
balanced_acc	semi_resampling	0.080	5.223e-03	0.069	0.090
balanced_acc	resampling_after_clustering	0.132	7.516e-03	0.117	0.147
mcc	no_resampling	0.046	6.500e-03	0.033	0.059
mcc	resampling_before_clustering	0.249	6.548e-03	0.236	0.261
mcc	semi_resampling	0.107	6.633e-03	0.094	0.120
mcc	resampling_after_clustering	0.282	9.546e-03	0.263	0.300

healthier distributions of predicted active ratios of `semi resampling` against `no resampling`, this made a **case in favour of the former**.

- In **five** out of five metrics, **proteins with more interactions were better predicted** (table S66).

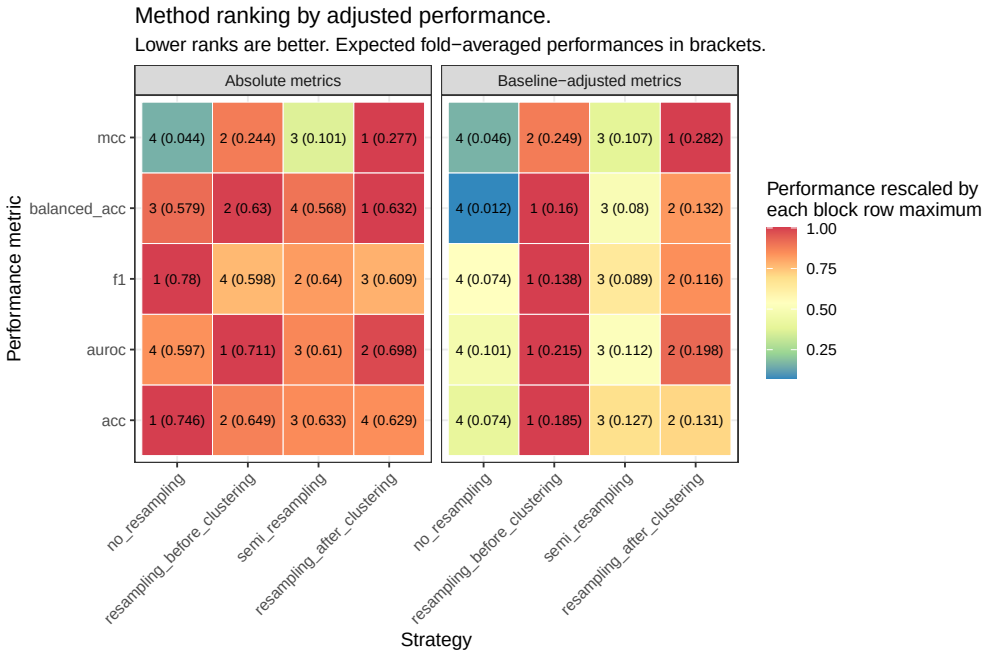

Figure S95: Method ranking according to the linear model predicted adjusted performances for each metric. Baseline metrics were taken into account in the adjustment. For a direct comparison, the same ranking using the absolute metrics was kept side by side.

C.2.6 Reproducibility

- R version 3.6.3 (2020-02-29), `x86_64-pc-linux-gnu`
- Locale: `LC_CTYPE=en_US.UTF-8`, `LC_NUMERIC=C`, `LC_TIME=es_ES.UTF-8`, `LC_COLLATE=en_US.UTF-8`, `LC_MONETARY=es_ES.UTF-8`, `LC_MESSAGES=en_US.UTF-8`, `LC_PAPER=es_ES.UTF-8`, `LC_NAME=C`, `LC_ADDRESS=C`, `LC_TELEPHONE=C`, `LC_MEASUREMENT=es_ES.UTF-8`, `LC_IDENTIFICATION=C`
- Running under: `Ubuntu 16.04.7 LTS`
- Matrix products: default
- BLAS: `/usr/lib/atlas-base/atlas/libblas.so.3.0`
- LAPACK: `/usr/lib/atlas-base/atlas/liblapack.so.3.0`
- Base packages: base, datasets, graphics, grDevices, methods, stats, utils
- Other packages: corrplot 0.84, dplyr 0.8.3, forcats 0.5.0, ggplot2 3.3.0, gsubfn 0.7, kableExtra 1.3.1, magrittr 1.5, proto 1.0.0, purrr 0.3.3, readr 1.3.1, rmarkdown 1.12, stargazer 5.2.2, stringr 1.4.0, tibble 2.1.3, tidyr 1.0.0, tidyverse 1.3.0
- Loaded via a namespace (and not attached): abind 1.4-5, assertthat 0.2.1, backports 1.1.4, bookdown 0.12, broom 0.5.3, car 3.0-10, carData 3.0-4, cellranger 1.1.0, cli 1.1.0, codetools 0.2-16, colorspace 1.4-1, compiler 3.6.3, crayon 1.3.4, curl 3.3, data.table 1.12.2,

DBI 1.0.0, dbplyr 1.4.2, digest 0.6.18, emmeans 1.5.3, estimability 1.3, evaluate 0.13, foreign 0.8-76, fs 1.3.1, generics 0.0.2, glue 1.3.1, grid 3.6.3, gtable 0.3.0, haven 2.2.0, highr 0.8, hms 0.5.3, htmltools 0.3.6, httr 1.4.1, jsonlite 1.6, knitr 1.22, labeling 0.3, lattice 0.20-41, lifecycle 0.1.0, lubridate 1.7.4, MASS 7.3-53, Matrix 1.2-18, mgcv 1.8-33, modelr 0.1.6, multcomp 1.4-15, munsell 0.5.0, mvtnorm 1.0-11, nlme 3.1-149, openxlsx 4.1.0.1, pillar 1.4.3, pkgconfig 2.0.2, plyr 1.8.4, R6 2.4.0, RColorBrewer 1.1-2, Rcpp 1.0.5, readxl 1.3.1, reprex 0.3.0, reshape2 1.4.3, rio 0.5.16, rlang 0.4.5, rstudioapi 0.10, rvest 0.3.5, sandwich 2.5-1, scales 1.0.0, splines 3.6.3, stringi 1.4.3, survival 3.2-7, tcltk 3.6.3, TH.data 1.0-10, tidyselect 0.2.5, tools 3.6.3, vctrs 0.2.4, viridisLite 0.3.0, webshot 0.5.2, withr 2.1.2, xfun 0.6, xml2 1.2.5, xtable 1.8-4, yaml 2.2.0, zip 2.0.4, zoo 1.8-6

C.3 APPENDIX S3 – MODEL PREDICTIONS AND PERFORMANCE (NRS)

C.3.1 Overview

This supplement describes the behaviour of the proteochemometrics (PCM) deep learning model to predict protein-compound bioactivity for **NRs**. Specifically, this includes the descriptive statistics of data imbalance: the proportion of actives per protein in the training and test sets during the model fitting and the predicted proportion of actives. The model performance per protein was also described, pinpointing the most influential factors and characterising the proteins with the most extreme performances.

Four strategies (no_resampling, resampling_before_clustering, semi_resampling, resampling_after_clustering) were considered. For each of those, 10 folds of repeated holdout were run, and 5 performance metrics were computed: acc, auroc, f1, balanced_acc, mcc. This led to a total of 1808 values of performance. Since some strategies involved the upsampling method SMOTE, proteins whose sample sizes did not allow upsampling were excluded (table S68).

Table S68: Number of proteins for which performance metrics were computed. The resampling after clustering was the most stringent strategy regarding eligible proteins, since the resampling was carried out after the clustering, which introduced more imbalance.

Strategy	Fold 0	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Fold 6	Fold 7	Fold 8	Fold 9
no_resampling	56	69	51	65	55	44	68	56	51	67
resampling_before_clustering	44	46	48	47	40	46	45	45	46	46
semi_resampling	56	69	51	65	55	44	68	56	51	67
resampling_after_clustering	19	20	19	25	16	17	20	14	21	20

C.3.2 Description of data balance

The data balancing strategy had an impact on the actual data balance, defined as the proportion of active molecules for a protein. Furthermore,

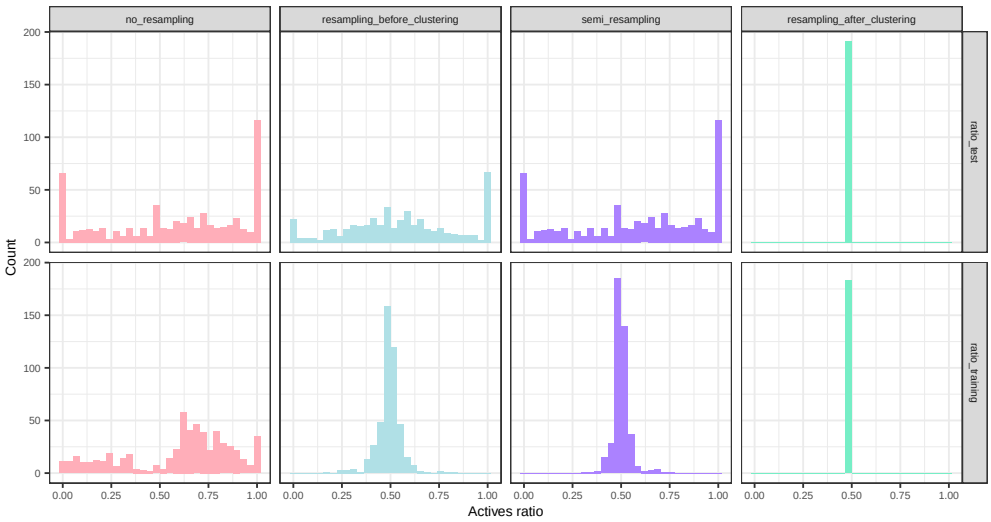

Figure S96: Distributios of the actives ratio in the training set and in the test set (both original and predicted by the deep learning model).

specific trends were observed in the original data in the training and test sets, as well as in the values predicted by the deep learning model.

Distributions of the actives ratio

The histograms in figure S96 revealed trends:

- `no_resampling` **keeps similar data imbalance in training and test.**
- `resampling_before_clustering` and `semi_resampling` **lead to a more balanced training set, but not so much for the test set.**
- `resampling_after_clustering` **kept balanced proteins** in both training and test sets.

In addition, **test sets with imbalance tended to magnify** it and create extreme cases (all actives or all inactives), probably due to the combination of the clustering and the lower sample sizes in the test sets compared to training.

Comparing training and test imbalance

Figure S97 revealed both positive, negative and null trends between the training and test set protein balances.

- `no_resampling` showed a **positive relation** between both, i.e. proteins were prone to keep their (im)balance in train and test.
- `resampling_before_clustering` showed an **inverse relationship** instead. This was expected since this strategy started from globally balanced proteins, and after the clustering, an imbalance in one direction in the training set entailed an inverse imbalance in the test set.
- `semi_resampling` led to **independent train and test balances, expected since the train set was resampled, breaking any correlation with the test set balance.**

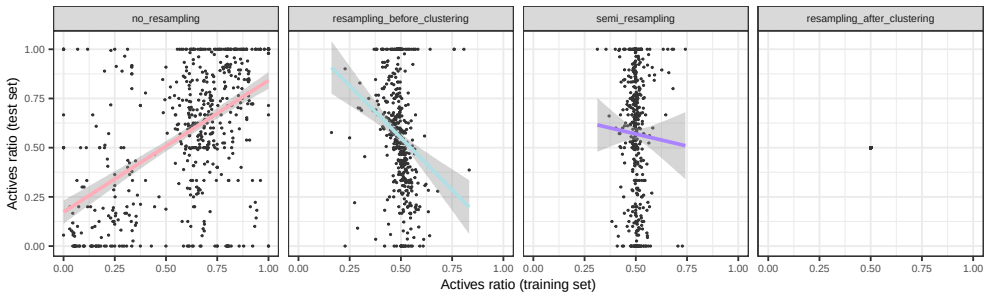

Figure S97: Comparison of the training and test ratios, by resampling strategy. A linear fit line was added per strategy.

- `resampling_after_clustering` always **kept balanced proteins**, by design.

Table S69 displays the Pearson correlation estimate, 95% confidence interval and p-value for each strategy (except `resampling_after_clustering`, where ratios are constant), further confirming the claims above.

Table S69: Correlations between train and test active ratios. 95% confidence intervals and p-values are shown.

strategy	cor	ci_lower	ci_upper	alternative	pvalue
no_resampling	0.531	0.470	0.588	two.sided	6.53e-43
resampling_before_clustering	-0.241	-0.326	-0.152	two.sided	2.16e-07
semi_resampling	-0.033	-0.128	0.062	two.sided	4.92e-01
resampling_after_clustering	NA	NA	NA	two.sided	NA

Other covariates

The effect of the number of interactions of each protein in its corresponding set and fold (figure S98) and the protein length in amino acids (figure S99) on the test set imbalance was investigated:

- **Proteins with greatest imbalance** (i.e. where `(0.5 - ratio_test)^2` was greatest) **tended to be among those with the least interactions. Linear correlations were significant** (table S70).
- **The sequence length had no obvious effect on the protein imbalance. Linear correlations were not significant** (`resampling_before_clustering`) or significant but low (`no_resampling`, `semi_resampling`).

Table S70: Correlations between imbalance (as defined above) and number of interactions. 95% confidence intervals and p-values are shown.

strategy	cor	ci_lower	ci_upper	alternative	pvalue
no_resampling	-0.396	-0.462	-0.325	two.sided	2.89e-23
resampling_before_clustering	-0.395	-0.470	-0.314	two.sided	2.28e-18
semi_resampling	-0.396	-0.462	-0.325	two.sided	2.89e-23
resampling_after_clustering	NA	NA	NA	two.sided	NA

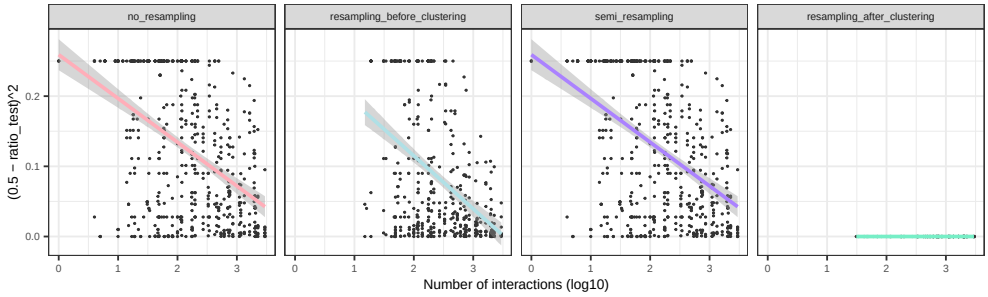

Figure S98: Data imbalance in the test set as a function of the number of available interactions for each protein.

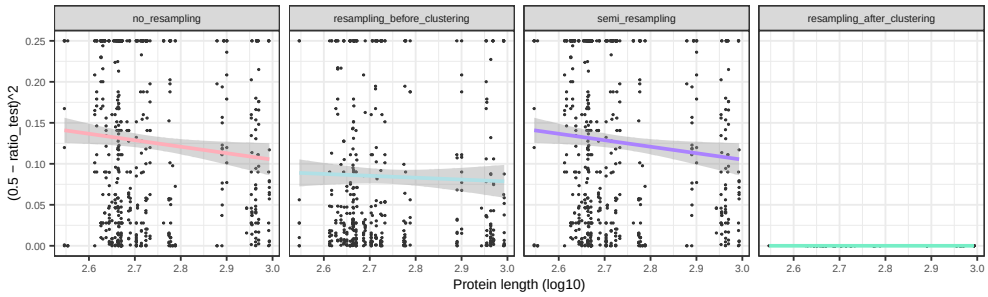

Figure S99: Data imbalance in the test set as a function of the protein length, in amino acids.

Table S71: Correlations between imbalance (as defined above) and sequence length. 95% confidence intervals and p-values are shown.

strategy	cor	ci_lower	ci_upper	alternative	pvalue
no_resampling	-0.091	-0.171	-0.010	two.sided	2.81e-02
resampling_before_clustering	-0.030	-0.122	0.062	two.sided	5.18e-01
semi_resampling	-0.091	-0.171	-0.010	two.sided	2.81e-02
resampling_after_clustering	NA	NA	NA	two.sided	NA

C.3.3 Linear models on predicted proportions

The next key question was to narrow down the factor driving the predicted proportion of actives. The main options under consideration were:

1. A constant, global imbalance that the model would learn from the whole dataset.
2. The protein-wise imbalance that the model would learn in the training set.
3. A test set-driven imbalance, based on its actual imbalance.

Distributions of the predicted ratios

After the model predictions in the test set were binarized (actives were those whose probabilities exceeded 0.5), the ratio of predicted actives was computed by protein. This ratio, shown in figure S100, suggested that:

- `no_resampling` was noticeably **biased to predict everything as positives**.

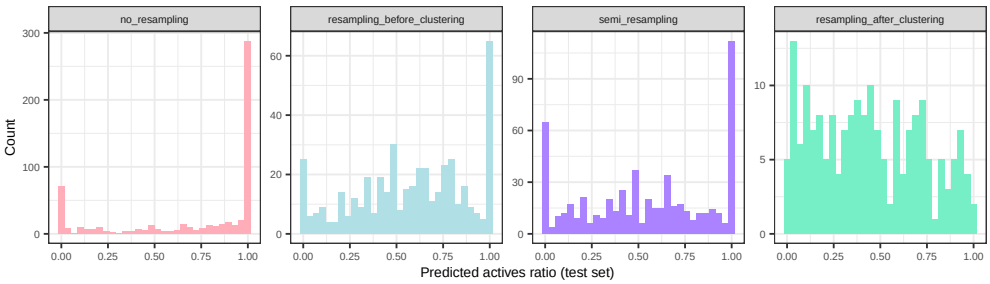

Figure S100: Ratios of the prediction values, after binarization.

- `resampling_before_clustering` and `semi_resampling` **alleviated the imbalance in the predictions, but still retained a spike of proteins where all the compounds were predicted as positives**.
- `resampling_after_clustering` **kept a wide and symmetric distribution of predicted actives**.

Now, representing together (1) the original training and test ratios, and (2) the predicted ratios in test (figure S101) eased a general qualitative assessment: **the distribution was most resemblant to that of the test proportions to that of the training ones** (except `resampling_after_clustering`, since those proportions are constant). Table S72 displays how `no_resampling` was **highly inclined to predict all positives**, `resampling_before_clustering` and `semi_resampling` **alleviated this phenomenon**, and `resampling_after_clustering` was **essentially balanced**.

Table S72: Percentage of extreme cases, i.e. proteins with all actives or inactives.

strategy	RatioSet	all_inactives	all_actives	all_extremes
no_resampling	ratio_test	11.3	19.6	30.9
no_resampling	ratio_test_predicted	12.2	47.6	59.8
no_resampling	ratio_training	1.8	6.1	7.9
resampling_before_clustering	ratio_test	4.9	14.8	19.6
resampling_before_clustering	ratio_test_predicted	5.5	14.1	19.6
resampling_before_clustering	ratio_training	0.0	0.0	0.0
semi_resampling	ratio_test	11.3	19.6	30.9
semi_resampling	ratio_test_predicted	11.2	19.2	30.4
semi_resampling	ratio_training	0.0	0.0	0.0
resampling_after_clustering	ratio_test	0.0	0.0	0.0
resampling_after_clustering	ratio_test_predicted	1.6	0.5	2.1
resampling_after_clustering	ratio_training	0.0	0.0	0.0

Predicted ratios against training ratios

Figure S102 puts the predicted ratios in context of the training ratios, elucidating a variety of trends:

- `no_resampling`: **positive trend between the training and the predicted ratio**, but since the training and the test ratio also positively correlated (figure 35), the latter could be the one driving the predicted ratio of positives.

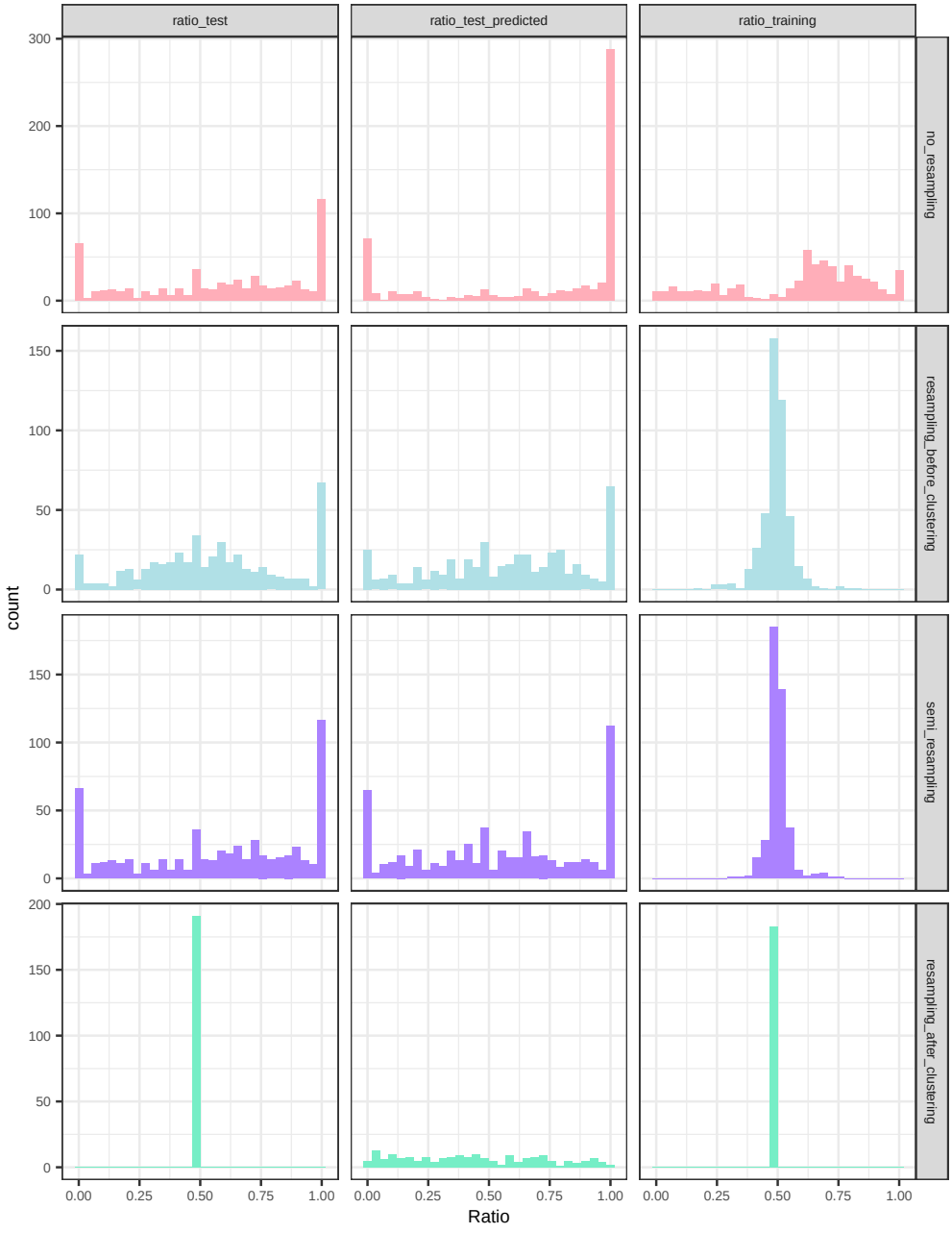

Figure S101: Distributios of the actives ratio in the training set and in the test set (both original and predicted by the deep learning model).

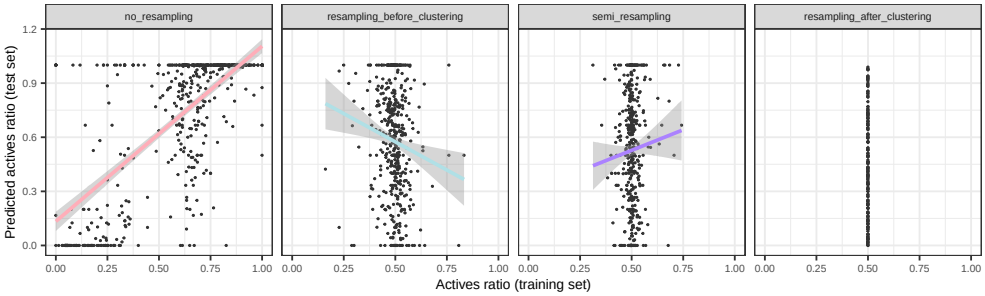

Figure S102: Predicted ratios, as a function of training ratios.

- `resampling_after_clustering` had a **constant training ratio**, meaning that the predicted ratio was not explainable by differences in training ratios.
- `resampling_before_clustering` showed instead a **negative relation between the training and the predicted ratio**. But since the former and the test ratio also anticorrelated (figure 35, the simplest explanation was that the test ratio drove the predicted test ratio.
- `semi_resampling` showed **no apparent correlation** between the predicted ratio and the training ratio.

The significance of the linear correlation backs up all the claims above (table S73).

Table S73: Correlations between train and predicted test active ratios. 95% confidence intervals and p-values are shown.

strategy	cor	ci_lower	ci_upper	alternative	pvalue
no_resampling	0.705	0.661	0.744	two.sided	5.22e-87
resampling_before_clustering	-0.136	-0.226	-0.045	two.sided	3.73e-03
semi_resampling	0.063	-0.032	0.158	two.sided	1.92e-01
resampling_after_clustering	NA	NA	NA	two.sided	NA

Linear models

The predicted ratio of actives r_{pred} was modelled through the following quasibinomial generalized linear models, stratified by strategy:

$$r_{pred} \sim r_{training} + r_{test} + \log10(n_{int}) + \log10(n_{seq}) + k_{fold}$$

The main variables of interest are the actual ratios in the training $r_{training}$ and in test r_{test}, both numeric between 0 and 1. As additional covariates, the number of interactions n_{int} and the sequence length n_{seq} (numerical) and the fold number k_{fold} (categorical) were also included.

IN SEMI_RESAMPLING OR RESAMPLING_BEFORE_CLUSTERING **Due to the lack of correlation between training and test ratios** (figure S97), the `semi_resampling` strategy is the **ideal scenario to disentangle their effects** on the predicted ratio of actives (see model in table S74). This additive model suggests:

- The **test ratio is driving the predicted proportions, rather than the training ratio**.
- `n_interactions`: **the term is not significant in NRs**.

Table S74 also shows the additive model for `resampling_before_clustering`. This strategy showed negative correlation between training and test ratios, also providing a reasonably good scenario to distinguish their effects.

- **This model confirms both conclusions from the model** in the `semi_resampling` strategy, with **similar estimates.**

Table S74: Linear models to describe the predicted active ratio for the semi_resampling and the resampling_before_clustering strategies. Significance and 95% confidence intervals are included.

	semi_resampling	resampling_before_clustering
	(1)	(2)
ratio_training	2.479 (−0.348, 5.307)	−1.249 (−3.03, 0.533)
	p = 0.086	p = 0.17
ratio_test	1.183 (0.798, 1.567)***	1.148 (0.722, 1.575)***
	p = 3.689e-09	p = 2.076e-07
log10(n_interactions)	−0.035 (−0.251, 0.18)	−0.104 (−0.308, 0.101)
	p = 0.748	p = 0.322
log10(len_seq)	2.467 (1.431, 3.504)**	1.306 (0.36, 2.251)*
	p = 4.167e-06	p = 7.074e-03
fold1	0.818 (0.292, 1.344)*	1.257 (0.731, 1.783)**
	p = 2.461e-03	p = 3.795e-06
fold2	0.104 (−0.412, 0.621)	0.356 (−0.131, 0.843)
	p = 0.692	p = 0.153
fold3	−0.315 (−0.84, 0.21)	0.906 (0.4, 1.412)**
	p = 0.24	p = 4.989e-04
fold4	0.359 (−0.153, 0.871)	0.347 (−0.169, 0.864)
	p = 0.17	p = 0.188
fold5	0.468 (−0.057, 0.992)	0.623 (0.126, 1.119)*
	p = 0.081	p = 0.014
fold6	0.373 (−0.129, 0.875)	0.305 (−0.197, 0.808)
	p = 0.146	p = 0.235
fold7	−0.186 (−0.686, 0.314)	0.776 (0.272, 1.28)*
	p = 0.466	p = 2.717e-03
fold8	0.51 (1.837e-03, 1.018)*	0.675 (0.174, 1.176)*
	p = 0.05	p = 8.564e-03
fold9	0.812 (0.288, 1.337)*	−0.069 (−0.564, 0.427)
	p = 2.557e-03	p = 0.786
Constant	−8.753 (−11.857, −5.65)***	−3.532 (−6.296, −0.769)*
	p = 5.744e-08	p = 0.013
Observations	425	451
Note:		*p<0.05; **p<1.000e-03; ***p<1e-06

IN NO_RESAMPLING The explanatory linear model under the `no_resampling` strategy (table S75) **suffers from the positive correlation between training and test ratios, which can be confounded**.

- Both `training_ratio` and `test_ratio` show a **positive effect** on the predicted fraction of actives.
- Although the **estimate is larger and more significant** for `training_ratio`, the **confounding effect and the very skewed distribution of the predicted ratios deems this model inconclusive**.

Conclusions

- Data imbalance exists in all strategies but in `resampling_after_clustering`, where balance is enforced.
- The correlation between a protein's ratio in train and test is positive in `no_resampling`, negative in `resampling_before_clustering` and null in `semi_resampling` and `resampling_after_clustering`.
- The main factor driving the ratio of actives in the model predictions, per protein, is the actual ratio of positives in the test set. Their distributions are resemblant, and linear models confirm the association.

All of them apply to NRs.

C.3.4 Description of baseline performance

Before evaluating the deep learning model, the performance metrics of the baselines were characterised, in order to pinpoint imbalance-sensitive and insensitive metrics. Metrics were called imbalance-sensitive if the imbalance-aware random baseline exhibited different performances between resampling strategies.

Descriptive plot

Figure S103 shows a fold-averaged picture of the metrics by protein. **Visual inspection suggested that accuracy, F1 and possibly balanced accuracy were affected by the data imbalance**. **F1 is the most apparent case**, see the quartiles in table S76.

Linear models

Formally, each performance metric was described with the following linear model:

$$\text{metric} \sim \text{strategy} + \log10(n_{\texttt{int}}) + \log10(n_{\texttt{seq}}) + k_{\texttt{fold}}$$

The response was the quantitative metric of interest (one model per metric), while `strategy` was categorical with the following possibilities: `no_resampling`, `resampling_after_clustering`, `resampling_before_clustering`, `semi_resampling`. Additional covariates included the number of interactions $n_{\texttt{int}}$ and the sequence length $n_{\texttt{seq}}$ (numerical) and the fold number $k_{\texttt{fold}}$ (categorical). The `strategy` variable was tested with a type 3 ANOVA, being **significant** with $p < 0.05$ for `acc`, `f1` and `balanced_acc` (table S77).

Based on this, metrics were divided in two types:

Table S75: Linear models to describe the predicted active ratio for the no_resampling strategy. Significance and 95% confidence intervals are included.

	no_resampling
ratio_training	5.556 (4.673, 6.44)***
	p = 4.823e-31
ratio_test	2.888 (2.242, 3.534)***
	p = 2.271e-17
log10(n_interactions)	−0.834 (−1.101, −0.568)***
	p = 1.616e-09
log10(len_seq)	1.202 (−0.471, 2.874)
	p = 0.16
fold1	0.24 (−0.591, 1.072)
	p = 0.571
fold2	−0.254 (−1.075, 0.568)
	p = 0.545
fold3	−1.286 (−2.06, −0.511)*
	p = 1.206e-03
fold4	0.322 (−0.603, 1.247)
	p = 0.495
fold5	0.077 (−0.791, 0.945)
	p = 0.862
fold6	−0.053 (−0.883, 0.778)
	p = 0.901
fold7	−0.632 (−1.465, 0.201)
	p = 0.138
fold8	−0.34 (−1.183, 0.503)
	p = 0.43
fold9	−0.384 (−1.186, 0.417)
	p = 0.348
Constant	−4.575 (−9.19, 0.039)
	p = 0.052
Observations	571
Note:	*p<0.05; **p<1.000e-03; ***p<1e-06

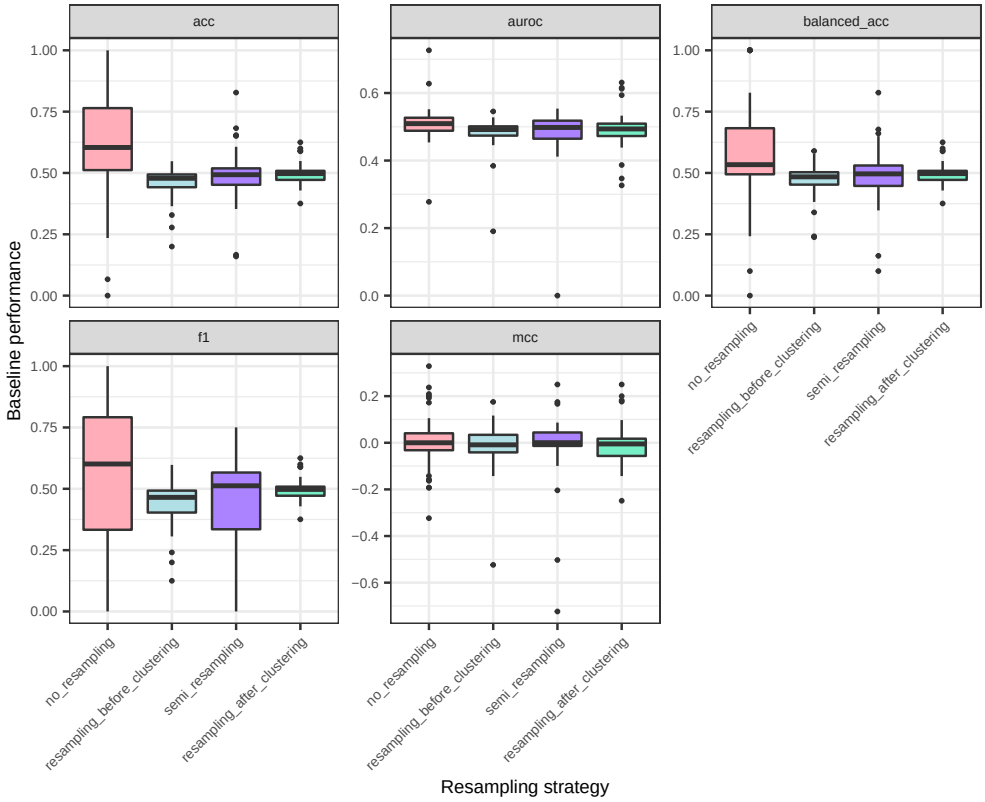

Figure S103: Performance metrics for imbalance-aware random baselines. Data points correspond to proteins, averaged over folds.

Table S76: Quartiles for the baseline F1-scores.

strategy	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
no_resampling	0.000	0.333	0.601	0.540	0.792	1.000
resampling_before_clustering	0.125	0.403	0.465	0.443	0.493	0.597
semi_resampling	0.000	0.335	0.512	0.446	0.566	0.751
resampling_after_clustering	0.376	0.472	0.497	0.497	0.509	0.625

Table S77: ANOVA p-values for including the resampling strategy as a regressor. Significant p-values imply that differences exist between resampling strategies.

strategy	variable	Sum Sq	Df	F value	Pr(>F)
acc	strategy	6.9011780	3	5.57e+01	2.74e-34
auroc	strategy	0.0137418	3	1.38e-01	9.38e-01
balanced_acc	strategy	2.1662131	3	1.59e+01	3.33e-10
f1	strategy	3.3010337	3	1.53e+01	7.73e-10
mcc	strategy	0.0466933	3	2.74e-01	8.44e-01

- Those where the baseline was different between strategies, i.e. imbalance-sensitive: `acc`, `f1` and `balanced_acc`. Therefore, before comparing strategies, the baseline performance needed to be accounted for.
- Those where the baseline was constant, i.e. imbalance-insensitive: `auroc`, `mcc`. Here we could compare strategies directly.

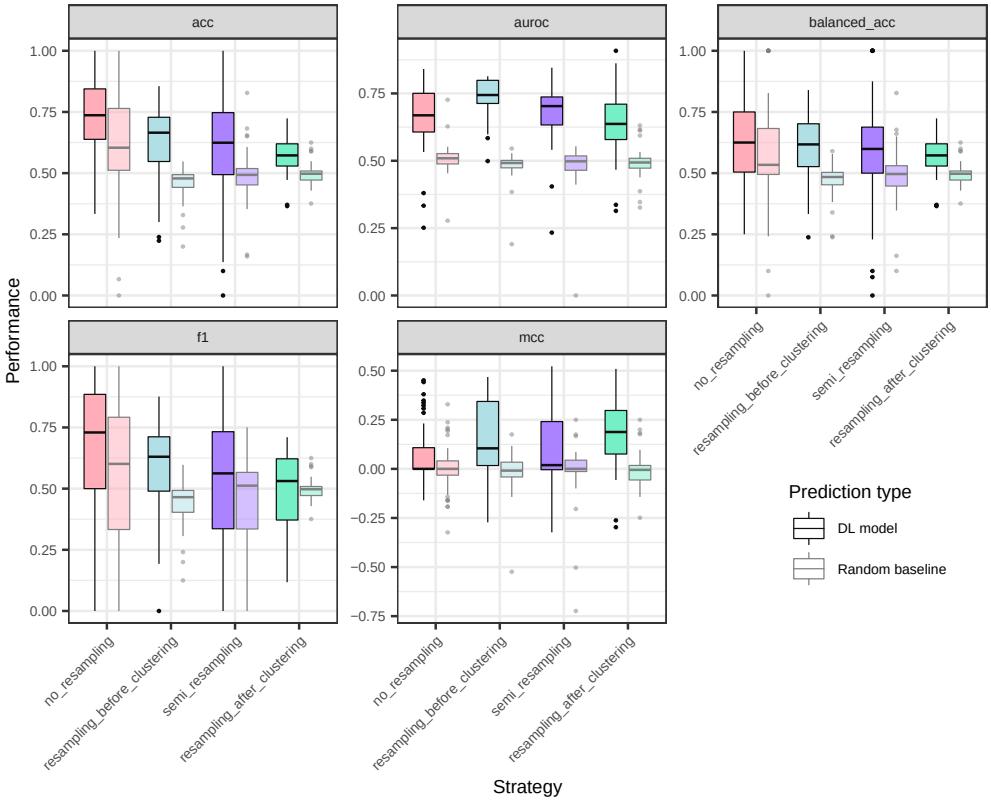

Figure S104: Performance metrics for balancing strategies and their corresponding imbalance-aware random baselines. Data points correspond to proteins, averaged over folds.

All applies to NRs as well.

C.3.5 Description of deep learning model performance

An overview of fold-averaged performances is displayed in figure S104, where strategies are paired with their baselines. This illustrates the **issue of direct strategy comparison** with imbalance-sensitive metrics, which was especially visible for the F1-score. Some metrics are undefined in edge cases (e.g. AUROC when only actives or only inactives are available); table S78 summarizes the number of proteins, added over folds, whose metrics were computable.

Table S78: Number of computable performance measures. AUROC was undefined for proteins with all actives or unactives in the test set, hence its lower counts.

strategy	acc	auroc	f1	balanced_acc	mcc
no_resampling	582	402	582	582	582
resampling_before_clustering	453	364	453	453	453
semi_resampling	582	402	582	582	582
resampling_after_clustering	191	191	191	191	191

Absolute, baseline-naive performance

Anologous to the baseline performance models, absolute metric models (not accounting for baselines) were fitted:

$$\text{metric} \sim \text{strategy} + \log10(n_{int}) + \log10(n_{seq}) + k_{fold}$$

The `strategy` covariate was **always significant** in a type 3 ANOVA (table S79). The models, summarized in S80, showed **different behaviour in imbalance-sensitive and insensitive metrics**. Pairwise comparisons of the strategy coefficients using Tukey's method would point to **two different pictures** (figure S105), further confirmed when prioritizing the strategies according to their expected performance through the linear models (table S81 and figure S106):

- **Accuracy, balanced accuracy and F1-score** suggested that `no_resampling` was the **best strategy**, but this was confounded by the fact that it also held the **highest baselines** .
- **AUROC and MCC** and showed instead that `resampling_before_clustering` held the **highest performance estimates**.

Table S79: ANOVA p-values for including the resampling strategy as a regressor in the performance models.

strategy	variable	Sum Sq	Df	F value	Pr(>F)
acc	strategy	5.8906886	3	3.01e+01	5.94e-19
auroc	strategy	0.6372396	3	3.79e+00	1.01e-02
f1	strategy	5.2009857	3	1.54e+01	6.59e-10
balanced_acc	strategy	0.8804841	3	4.74e+00	2.70e-03
mcc	strategy	1.1138090	3	5.16e+00	1.50e-03

Table S80: Linear models to describe each performance metric. Standard deviations in parentheses.

	acc	auroc	f1	balanced_acc	mcc
	(1)	(2)	(3)	(4)	(5)
strategyresampling_before_clustering	−0.101***	0.032	−0.054*	−0.019	0.065**
	(0.016)	(0.017)	(0.021)	(0.016)	(0.017)
	p = 6.966e-10	p = 0.067	p = 0.011	p = 0.23	p = 1.307e-04
strategysemi_resampling	−0.114***	−0.016	−0.105***	−0.047*	0.033*
	(0.015)	(0.017)	(0.02)	(0.015)	(0.016)
	p = 4.598e-14	p = 0.335	p = 1.075e-07	p = 1.207e-03	p = 0.034
strategyresampling_after_clustering	−0.171***	−0.03	−0.166***	−0.06*	0.017
	(0.022)	(0.021)	(0.029)	(0.022)	(0.023)
	p = 3.743e-14	p = 0.156	p = 1.699e-08	p = 6.135e-03	p = 0.475
log10(n_interactions)	0.035**	0.114***	0.057**	0.015	0.131***
	(9.064e-03)	(0.011)	(0.012)	(8.828e-03)	(9.513e-03)
	p = 1.451e-04	p = 1.867e-25	p = 1.952e-06	p = 0.088	p = 5.040e-41
log10(len_seq)	−0.047	0.178**	0.258**	−0.077	0.074
	(0.051)	(0.053)	(0.066)	(0.049)	(0.053)
	p = 0.348	p = 7.888e-04	p = 1.033e-04	p = 0.118	p = 0.163
fold1	0.091**	7.686e-03	0.174***	0.048	0.025
	(0.026)	(0.028)	(0.035)	(0.026)	(0.028)
	p = 5.510e-04	p = 0.782	p = 5.102e-07	p = 0.063	p = 0.361
fold2	−4.926e-03	0.028	0.022	−0.019	0.03
	(0.028)	(0.029)	(0.036)	(0.027)	(0.029)
	p = 0.858	p = 0.326	p = 0.546	p = 0.478	p = 0.295
fold3	0.021	0.041	−0.013	0.016	0.036
	(0.026)	(0.028)	(0.035)	(0.026)	(0.028)
	p = 0.43	p = 0.148	p = 0.704	p = 0.535	p = 0.2
fold4	0.068*	−1.440e-03	0.143**	0.055*	−0.02
	(0.028)	(0.031)	(0.036)	(0.027)	(0.029)
	p = 0.014	p = 0.962	p = 8.640e-05	p = 0.042	p = 0.496
fold5	0.017	−0.026	−0.013	0.013	−0.03
	(0.028)	(0.03)	(0.037)	(0.028)	(0.03)
	p = 0.542	p = 0.39	p = 0.731	p = 0.641	p = 0.308
fold6	0.061*	0.019	0.137**	0.033	0.024
	(0.026)	(0.028)	(0.035)	(0.026)	(0.028)
	p = 0.02	p = 0.49	p = 8.201e-05	p = 0.195	p = 0.379
fold7	0.045	0.032	0.066	0.033	0.026
	(0.027)	(0.03)	(0.036)	(0.027)	(0.029)
	p = 0.1	p = 0.285	p = 0.067	p = 0.223	p = 0.364
fold8	0.09*	0.036	0.1*	0.073*	0.059*
	(0.028)	(0.029)	(0.036)	(0.027)	(0.029)
	p = 1.066e-03	p = 0.224	p = 5.932e-03	p = 6.518e-03	p = 0.043
fold9	0.097**	0.047	0.143**	0.074*	0.075*
	(0.027)	(0.028)	(0.035)	(0.026)	(0.028)
	p = 2.477e-04	p = 0.091	p = 3.981e-05	p = 4.380e-03	p = 6.852e-03
Constant	0.738***	−0.155	−0.28	0.781***	−0.423*
	(0.139)	(0.145)	(0.183)	(0.136)	(0.146)
	p = 1.291e-07	p = 0.286	p = 0.125	p = 9.610e-09	p = 3.832e-03
Observations	1808	1359	1808	1808	1808
R^2	0.068	0.107	0.078	0.023	0.129
Adjusted R^2	0.061	0.098	0.071	0.015	0.122

Note: *p<0.05; **p<1.000e-03; ***p<1e-06

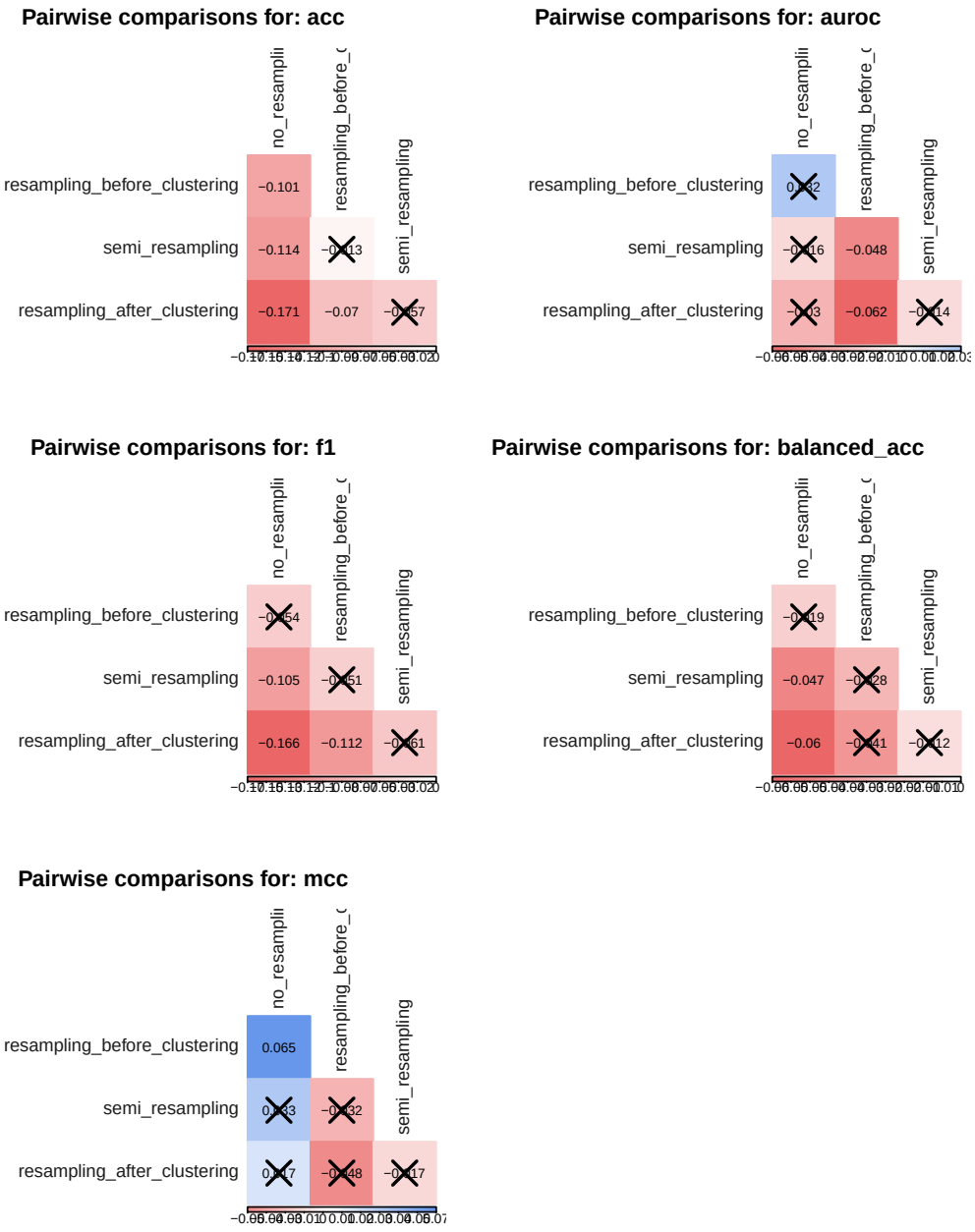

Figure S105: Pairwise comparison of strategy performance using Tukey method.

Table S81: Expected absolute performances, by metric and strategy, with 95% confidence intervals.

metric	strategy	emmean	SE	lower.CL	upper.CL
acc	no_resampling	0.750	1.189e-02	0.727	0.773
acc	resampling_before_clustering	0.649	1.238e-02	0.625	0.674
acc	semi_resampling	0.636	1.189e-02	0.613	0.659
acc	resampling_after_clustering	0.579	1.857e-02	0.543	0.616
auroc	no_resampling	0.664	1.237e-02	0.640	0.688
auroc	resampling_before_clustering	0.695	1.263e-02	0.671	0.720
auroc	semi_resampling	0.648	1.237e-02	0.623	0.672
auroc	resampling_after_clustering	0.633	1.723e-02	0.600	0.667
f1	no_resampling	0.664	1.560e-02	0.633	0.695
f1	resampling_before_clustering	0.610	1.624e-02	0.578	0.642
f1	semi_resampling	0.559	1.560e-02	0.528	0.590
f1	resampling_after_clustering	0.498	2.435e-02	0.450	0.546
balanced_acc	no_resampling	0.642	1.158e-02	0.619	0.665
balanced_acc	resampling_before_clustering	0.623	1.206e-02	0.600	0.647
balanced_acc	semi_resampling	0.595	1.158e-02	0.572	0.618
balanced_acc	resampling_after_clustering	0.582	1.808e-02	0.547	0.618
mcc	no_resampling	0.162	1.248e-02	0.138	0.187
mcc	resampling_before_clustering	0.227	1.300e-02	0.202	0.253
mcc	semi_resampling	0.195	1.248e-02	0.171	0.220
mcc	resampling_after_clustering	0.179	1.949e-02	0.141	0.217

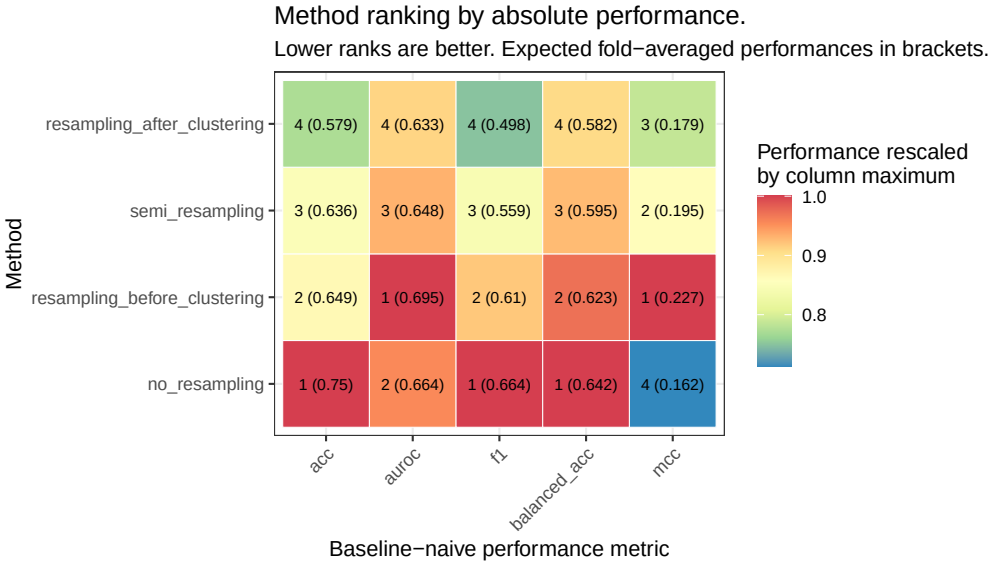

Figure S106: Method ranking according to the linear model predicted performances for each metric. Baseline metrics were ignored.

Baseline-adjusted performance

To address the pitfalls of the direct comparison of metrics whose baselines may differ, baseline-adjusted performance metrics were defined and modelled analogously. Specifically:

$$\text{adj\_metric} = \text{metric} - \text{baseline}$$

A descriptive plot of the the adjusted metrics (figure S107) pointed to a **scenario diffent than that of unadjusted ones** (figure S104).

Adjusted performance metrics were described with the following linear model:

$$\text{adj\_metric} \sim \text{strategy} + \log10(n_{int}) + \log10(n_{seq}) + k_{fold}$$

Note that while all metrics but `mcc` were non-negative, the adjusted metrics could show negative values when the performance of the DL model was lower than that of the baseline.

Again, `strategy` covariate was **always significant except for AUROC in NRs** in a type 3 ANOVA (table S82). **Baseline adjustment brought a uniform behaviour across the models** (table S83), further confirmed in pairwise coefficient comparison (Tukey's method, figure S108) and in their expected performance (table S84 and figure S109):

- **In NRs**, `resampling_before_clustering` **was the best performer, followed by a technical tie between** `no_resampling` and `semi_resampling`, and `resampling_after_clustering`.
- **In NRs the statistical significance is lower, but the general picture in the imbalance-insensitve non-adjusted metrics remains in the adjusted metrics: resampling before clustering being the best performer and no resampling being sensibly penalised** .

Table S82: ANOVA p-values for including the resampling strategy as a regressor in the adjusted performance models.

strategy	variable	Sum Sq	Df	F value	Pr(>F)
acc	strategy	1.4366294	3	6.41e+00	2.57e-04
auroc	strategy	0.2707975	3	1.14e+00	3.31e-01
f1	strategy	2.2855264	3	1.03e+01	1.05e-06
balanced_acc	strategy	1.4034456	3	6.11e+00	3.96e-04
mcc	strategy	1.4303470	3	3.57e+00	1.36e-02

Conclusions drawn from the baseline-adjusted performance analysis:

- **In NRs, resampling after clustering and semi resampling are technically tied, so data augmentation in the test set is not the largest performance drive**.
- **In NRs, the almost six-fold reduction in number of samples compared to kinases probably limited the statistical power when comparing semi resampling and no resampling**. Tukey's method gave

Table S83: Linear models to describe each adjusted performance metric. Standard deviations in parentheses.

	acc	auroc	f1	balanced_acc	mcc
	(1)	(2)	(3)	(4)	(5)
strategyresampling_before_clustering	0.056*	0.027	0.05*	0.072**	0.074*
	(0.017)	(0.021)	(0.017)	(0.018)	(0.023)
	p = 1.474e-03	p = 0.182	p = 4.337e-03	p = 4.886e-05	p = 1.637e-03
strategysemi_resampling	0.028	6.049e-03	−3.269e-03	0.033	0.051*
	(0.018)	(0.021)	(0.018)	(0.018)	(0.024)
	p = 0.111	p = 0.774	p = 0.854	p = 0.067	p = 0.033
strategyresampling_after_clustering	−0.038	−0.017	−0.084**	6.519e-03	0.033
	(0.025)	(0.026)	(0.025)	(0.025)	(0.033)
	p = 0.121	p = 0.516	p = 6.993e-04	p = 0.795	p = 0.323
log10(n_interactions)	0.037*	0.093***	0.025*	0.035*	0.137***
	(0.011)	(0.014)	(0.011)	(0.011)	(0.015)
	p = 1.227e-03	p = 1.128e-10	p = 0.026	p = 2.006e-03	p = 4.000e-19
log10(len_seq)	−4.762e-03	0.133*	0.098	0.023	0.059
	(0.056)	(0.065)	(0.056)	(0.057)	(0.076)
	p = 0.933	p = 0.039	p = 0.082	p = 0.691	p = 0.434
fold1	0.099**	0.024	0.139**	0.068*	0.059
	(0.03)	(0.034)	(0.03)	(0.03)	(0.04)
	p = 8.824e-04	p = 0.484	p = 2.766e-06	p = 0.024	p = 0.138
fold2	−1.428e-04	8.102e-03	0.028	−0.015	0.037
	(0.03)	(0.035)	(0.03)	(0.031)	(0.041)
	p = 0.996	p = 0.817	p = 0.349	p = 0.633	p = 0.369
fold3	0.046	0.028	0.03	0.046	0.054
	(0.03)	(0.035)	(0.03)	(0.03)	(0.04)
	p = 0.125	p = 0.425	p = 0.307	p = 0.128	p = 0.178
fold4	0.054	−0.021	0.071*	0.038	−0.012
	(0.031)	(0.037)	(0.031)	(0.031)	(0.041)
	p = 0.078	p = 0.574	p = 0.021	p = 0.222	p = 0.767
fold5	0.035	−0.06	0.048	0.031	−0.032
	(0.031)	(0.036)	(0.031)	(0.032)	(0.042)
	p = 0.259	p = 0.099	p = 0.12	p = 0.329	p = 0.438
fold6	0.05	0.019	0.081*	0.036	0.044
	(0.03)	(0.035)	(0.03)	(0.03)	(0.04)
	p = 0.092	p = 0.591	p = 6.661e-03	p = 0.235	p = 0.275
fold7	0.037	4.122e-03	0.05	0.026	0.045
	(0.03)	(0.036)	(0.03)	(0.031)	(0.041)
	p = 0.219	p = 0.909	p = 0.1	p = 0.398	p = 0.263
fold8	0.092*	0.036	0.109**	0.091*	0.095*
	(0.03)	(0.035)	(0.03)	(0.031)	(0.041)
	p = 2.382e-03	p = 0.303	p = 3.287e-04	p = 3.253e-03	p = 0.02
fold9	0.083*	0.049	0.093*	0.07*	0.1*
	(0.03)	(0.034)	(0.03)	(0.03)	(0.04)
	p = 5.452e-03	p = 0.151	p = 1.774e-03	p = 0.022	p = 0.012
Constant	−0.014	−0.471*	−0.31*	−0.121	−0.422*
	(0.155)	(0.177)	(0.155)	(0.157)	(0.207)
	p = 0.928	p = 7.935e-03	p = 0.045	p = 0.441	p = 0.042
Observations	1630	1277	1630	1630	1630
R^2	0.031	0.049	0.042	0.03	0.077
Adjusted R^2	0.022	0.039	0.033	0.022	0.069

Note: *p<0.05; **p<1.000e-03; ***p<1e-06

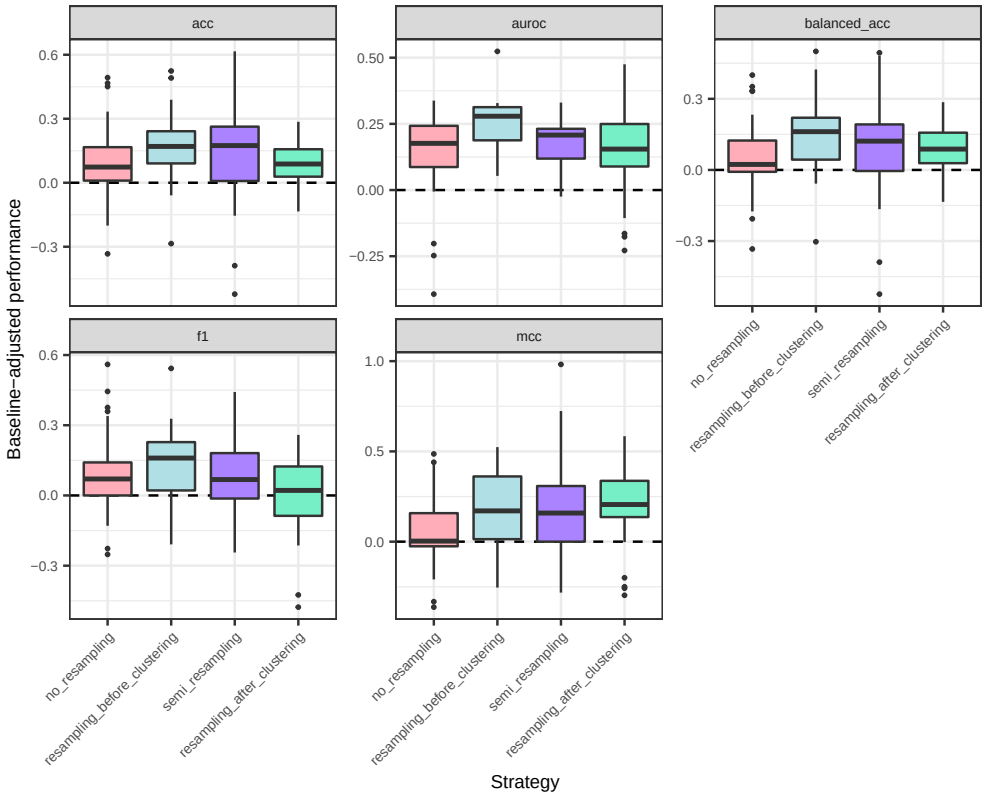

Figure S107: Baseline-adjusted performance metrics for balancing strategies. Data points correspond to proteins, averaged over folds.

Table S84: Expected adjusted performances, by metric and strategy, with 95% confidence intervals.

metric	strategy	emmean	SE	lower.CL	upper.CL
acc	no_resampling	0.124	1.338e-02	0.098	0.151
acc	resampling_before_clustering	0.180	1.353e-02	0.153	0.207
acc	semi_resampling	0.153	1.376e-02	0.126	0.180
acc	resampling_after_clustering	0.086	2.029e-02	0.046	0.126
auroc	no_resampling	0.163	1.507e-02	0.134	0.193
auroc	resampling_before_clustering	0.191	1.520e-02	0.161	0.221
auroc	semi_resampling	0.169	1.564e-02	0.139	0.200
auroc	resampling_after_clustering	0.146	2.090e-02	0.105	0.187
f1	no_resampling	0.095	1.333e-02	0.069	0.121
f1	resampling_before_clustering	0.145	1.348e-02	0.118	0.171
f1	semi_resampling	0.092	1.370e-02	0.065	0.119
f1	resampling_after_clustering	0.011	2.021e-02	-0.028	0.051
balanced_acc	no_resampling	0.079	1.356e-02	0.053	0.106
balanced_acc	resampling_before_clustering	0.151	1.371e-02	0.124	0.178
balanced_acc	semi_resampling	0.112	1.393e-02	0.085	0.140
balanced_acc	resampling_after_clustering	0.086	2.055e-02	0.045	0.126
mcc	no_resampling	0.161	1.790e-02	0.126	0.196
mcc	resampling_before_clustering	0.234	1.810e-02	0.199	0.270
mcc	semi_resampling	0.212	1.840e-02	0.176	0.248
mcc	resampling_after_clustering	0.194	2.713e-02	0.140	0.247

no significant differences at $p < 0.05$. However, the model coefficient of semi resampling was significantly greater than the reference level

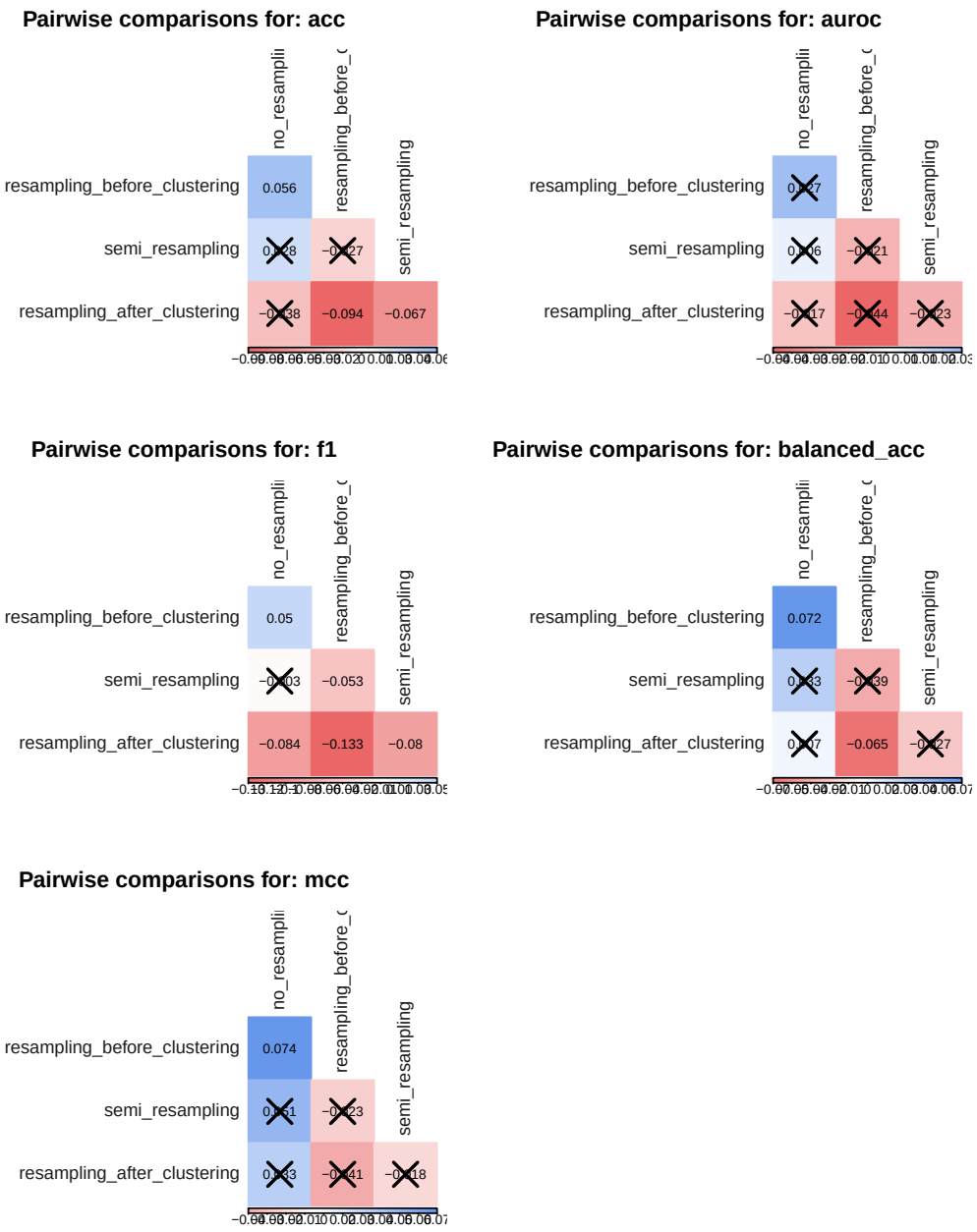

Figure S108: Pairwise comparison of strategy adjusted performance using Tukey method.

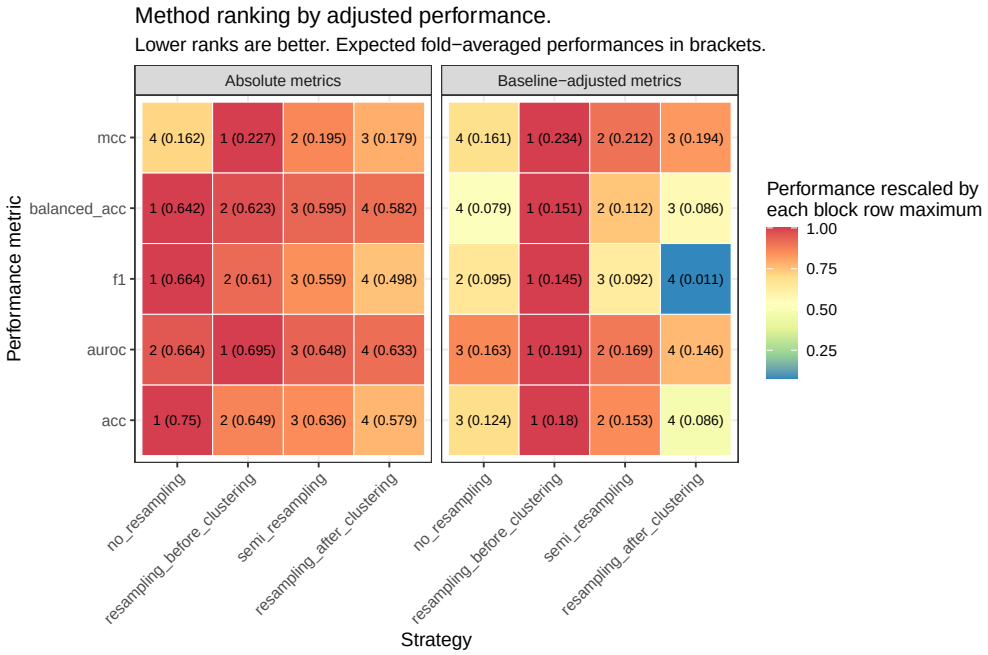

Figure S109: Method ranking according to the linear model predicted adjusted performances for each metric. Baseline metrics were taken into account in the adjustment. For a direct comparison, the same ranking using the absolute metrics was kept side by side.

(no resampling) in MCC ($p = 0.033$), but not in balanced accuracy ($p = 0.067$) or accuracy ($p = 0.111$), despite the positive estimates}, see table S83. This was **consistent with the observation that the predicted proportion of positives of the PCM model was mainly driven by the actual data balance in the test set, rather than that of the training set. Combined with the healthier distributions of predicted active ratios** of `semi_resampling` against `no_resampling`, this made a **case in favour of the former**.

- In **five** out of five metrics, **proteins with more interactions were better predicted** (table S83).

c.3.6 Reproducibility

- R version 3.6.3 (2020-02-29), `x86_64-pc-linux-gnu`
- Locale: `LC_CTYPE=en_US.UTF-8`, `LC_NUMERIC=C`, `LC_TIME=es_ES.UTF-8`, `LC_COLLATE=en_US.UTF-8`, `LC_MONETARY=es_ES.UTF-8`, `LC_MESSAGES=en_US.UTF-8`, `LC_PAPER=es_ES.UTF-8`, `LC_NAME=C`, `LC_ADDRESS=C`, `LC_TELEPHONE=C`, `LC_MEASUREMENT=es_ES.UTF-8`, `LC_IDENTIFICATION=C`
- Running under: `Ubuntu 16.04.7 LTS`
- Matrix products: default
- BLAS: `/usr/lib/atlas-base/atlas/libblas.so.3.0`
- LAPACK: `/usr/lib/atlas-base/atlas/liblapack.so.3.0`

- Base packages: base, datasets, graphics, grDevices, methods, stats, utils

- Other packages: corrplot 0.84, dplyr 0.8.3, forcats 0.5.0, ggplot2 3.3.0, gsubfn 0.7, kableExtra 1.3.1, magrittr 1.5, proto 1.0.0, purrr 0.3.3, readr 1.3.1, rmarkdown 1.12, stargazer 5.2.2, stringr 1.4.0, tibble 2.1.3, tidyr 1.0.0, tidyverse 1.3.0

- Loaded via a namespace (and not attached): abind 1.4-5, assertthat 0.2.1, backports 1.1.4, bookdown 0.12, broom 0.5.3, car 3.0-10, carData 3.0-4, cellranger 1.1.0, cli 1.1.0, codetools 0.2-16, colorspace 1.4-1, compiler 3.6.3, crayon 1.3.4, curl 3.3, data.table 1.12.2, DBI 1.0.0, dbplyr 1.4.2, digest 0.6.18, emmeans 1.5.3, estimability 1.3, evaluate 0.13, foreign 0.8-76, fs 1.3.1, generics 0.0.2, glue 1.3.1, grid 3.6.3, gtable 0.3.0, haven 2.2.0, highr 0.8, hms 0.5.3, htmltools 0.3.6, httr 1.4.1, jsonlite 1.6, knitr 1.22, labeling 0.3, lattice 0.20-41, lifecycle 0.1.0, lubridate 1.7.4, MASS 7.3-53, Matrix 1.2-18, mgcv 1.8-33, modelr 0.1.6, multcomp 1.4-15, munsell 0.5.0, mvtnorm 1.0-11, nlme 3.1-149, openxlsx 4.1.0.1, pillar 1.4.3, pkgconfig 2.0.2, plyr 1.8.4, R6 2.4.0, RColorBrewer 1.1-2, Rcpp 1.0.5, readxl 1.3.1, reprex 0.3.0, reshape2 1.4.3, rio 0.5.16, rlang 0.4.5, rstudioapi 0.10, rvest 0.3.5, sandwich 2.5-1, scales 1.0.0, splines 3.6.3, stringi 1.4.3, survival 3.2-7, tcltk 3.6.3, TH.data 1.0-10, tidyselect 0.2.5, tools 3.6.3, vctrs 0.2.4, viridisLite 0.3.0, webshot 0.5.2, withr 2.1.2, xfun 0.6, xml2 1.2.5, xtable 1.8-4, yaml 2.2.0, zip 2.0.4, zoo 1.8-6

C.4 APPENDIX S4 – MODEL PREDICTIONS AND PERFORMANCE (PROTEASES)

C.4.1 Overview

This supplement describes the behaviour of the proteochemometrics (PCM) deep learning model to predict protein-compound bioactivity for **PRs**. Specifically, this includes the descriptive statistics of data imbalance: the proportion of actives per protein in the training and test sets during the model fitting and the predicted proportion of actives. The model performance per protein was also described, pinpointing the most influential factors and characterising the proteins with the most extreme performances.

Four strategies (no_resampling, resampling_before_clustering, semi_resampling, resampling_after_clustering) were considered. For each of those, 10 folds of repeated holdout were run, and 5 performance metrics were computed: acc, auroc, f1, balanced_acc, mcc. This led to a total of 5904 values of performance. Since some strategies involved the upsampling method SMOTE, proteins whose sample sizes did not allow upsampling were excluded (table S85).

Table S85: Number of proteins for which performance metrics were computed. The resampling after clustering was the most stringent strategy regarding eligible proteins, since the resampling was carried out after the clustering, which introduced more imbalance.

Strategy	Fold 0	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Fold 6	Fold 7	Fold 8	Fold 9
no_resampling	187	178	210	198	206	171	184	192	205	165
resampling_before_clustering	144	144	141	149	154	152	149	140	150	150
semi_resampling	187	178	210	198	206	171	184	192	205	165
resampling_after_clustering	63	53	74	62	75	57	59	68	71	57

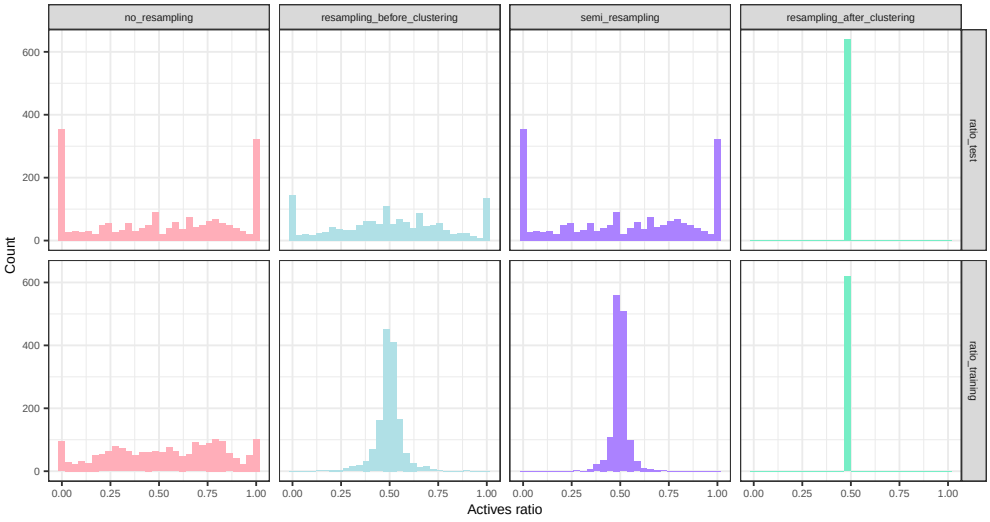

Figure S110: Distribution of the actives ratio in the training set and in the test set (both original and predicted by the deep learning model).

C.4.2 Description of data balance

The data balancing strategy had an impact on the actual data balance, defined as the proportion of active molecules for a protein. Furthermore, specific trends were observed in the original data in the training and test sets, as well as in the values predicted by the deep learning model.

Distributions of the actives ratio

The histograms in figure S110 revealed trends:

- `no_resampling` **keeps similar data imbalance in training and test.**
- `resampling_before_clustering` and `semi_resampling` **lead to a more balanced training set, but not so much for the test set.**
- `resampling_after_clustering` **kept balanced proteins** in both training and test sets.

In addition, **test sets with imbalance tended to magnify** it and create extreme cases (all actives or all inactives), probably due to the combination of the clustering and the lower sample sizes in the test sets compared to training.

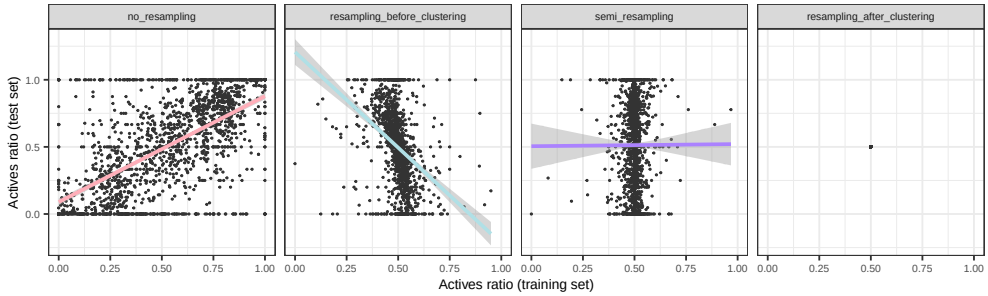

Figure S111: Comparison of the training and test ratios, by resampling strategy. A linear fit line was added per strategy.

Comparing training and test imbalance

Figure S111 revealed both positive, negative and null trends between the training and test set protein balances.

- `no_resampling` showed a **positive relation** between both, i.e. proteins were prone to keep their (im)balance in train and test.
- `resampling_before_clustering` showed an **inverse relationship** instead. This was expected since this strategy started from globally balanced proteins, and after the clustering, an imbalance in one direction in the training set entailed an inverse imbalance in the test set.
- `semi_resampling` led to **independent train and test balances, expected since the train set was resampled, breaking any correlation with the test set balance.**
- `resampling_after_clustering` always **kept balanced proteins**, by design.

Table S86 displays the Pearson correlation estimate, 95% confidence interval and p-value for each strategy (except `resampling_after_clustering`, where ratios are constant), further confirming the claims above.

Table S86: Correlations between train and test active ratios. 95% confidence intervals and p-values are shown.

strategy	cor	ci_lower	ci_upper	alternative	pvalue
no_resampling	0.630	0.602	0.657	two.sided	5.40e-206
resampling_before_clustering	-0.359	-0.403	-0.314	two.sided	1.58e-45
semi_resampling	0.002	-0.050	0.055	two.sided	9.26e-01
resampling_after_clustering	NA	NA	NA	two.sided	NA

Other covariates

The effect of the number of interactions of each protein in its corresponding set and fold (figure S112) and the protein length in amino acids (figure S113) on the test set imbalance was investigated:

- **Proteins with greatest imbalance** (i.e. where `(0.5 - ratio_test)^2` was greatest) **tended to be among those with the least interactions. Linear correlations were significant** (table S87).

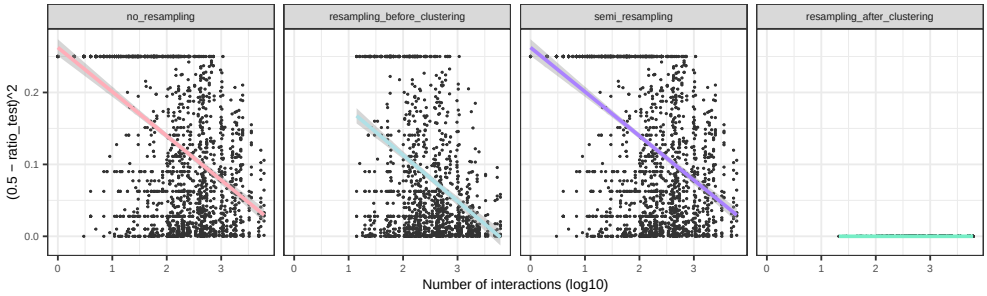

Figure S112: Data imbalance in the test set as a function of the number of available interactions for each protein.

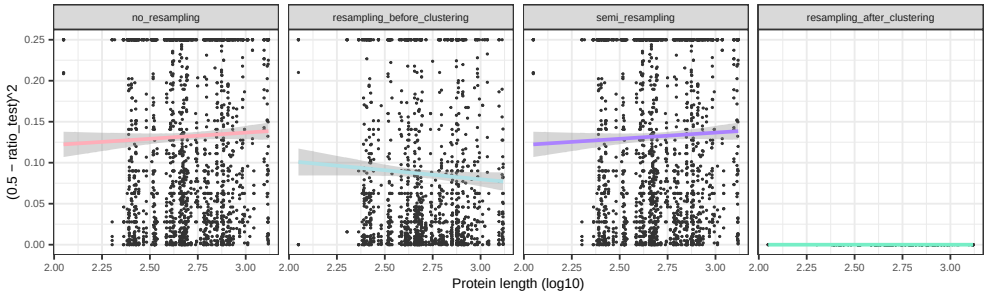

Figure S113: Data imbalance in the test set as a function of the protein length, in amino acids.

- **The sequence length had no obvious effect on the protein imbalance. Linear correlations were not significant** (`resampling_before_clustering`) or significant but low (`no_resampling`, `semi_resampling`).

Table S87: Correlations between imbalance (as defined above) and number of interactions. 95% confidence intervals and p-values are shown.

strategy	cor	ci_lower	ci_upper	alternative	pvalue
no_resampling	-0.289	-0.330	-0.247	two.sided	7.19e-38
resampling_before_clustering	-0.294	-0.340	-0.246	two.sided	2.81e-30
semi_resampling	-0.289	-0.330	-0.247	two.sided	7.19e-38
resampling_after_clustering	NA	NA	NA	two.sided	NA

Table S88: Correlations between imbalance (as defined above) and sequence length. 95% confidence intervals and p-values are shown.

strategy	cor	ci_lower	ci_upper	alternative	pvalue
no_resampling	0.049	0.004	0.094	two.sided	3.15e-02
resampling_before_clustering	-0.035	-0.086	0.016	two.sided	1.81e-01
semi_resampling	0.049	0.004	0.094	two.sided	3.15e-02
resampling_after_clustering	NA	NA	NA	two.sided	NA

C.4.3 Linear models on predicted proportions

The next key question was to narrow down the factor driving the predicted proportion of actives. The main options under consideration were:

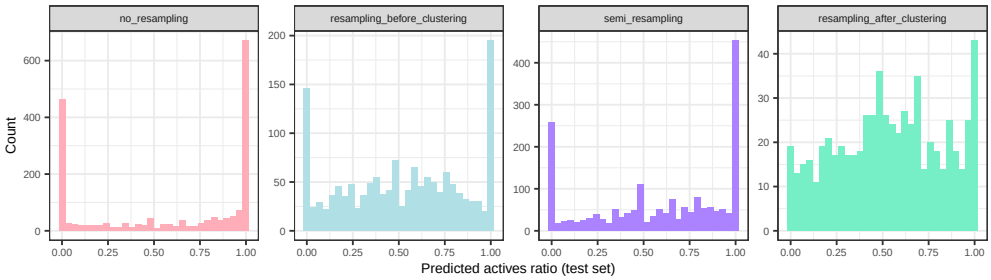

Figure S114: Ratios of the prediction values, after binarization.

1. A constant, global imbalance that the model would learn from the whole dataset.
2. The protein-wise imbalance that the model would learn in the training set.
3. A test set-driven imbalance, based on its actual imbalance.

Distributions of the predicted ratios

After the model predictions in the test set were binarized (actives were those whose probabilities exceeded 0.5), the ratio of predicted actives was computed by protein. This ratio, shown in figure S114, suggested that:

- `no_resampling` was noticeably **inclined to predict more positives than negatives, and in general more extreme ratios than the real ones**.
- `resampling_before_clustering` and `semi_resampling` **alleviated the imbalance in the predictions, being was quite close to the actual distribution**.
- `resampling_after_clustering` **kept a wide and symmetric distribution of predicted actives**.

Now, representing together (1) the original training and test ratios, and (2) the predicted ratios in test (figure S115) eased a general qualitative assessment: **the distribution was most resemblant to that of the test proportions to that of the training ones** (except `resampling_after_clustering`, since those proportions are constant). Table S89 displays how `no_resampling` was **inclined to predict more extreme cases (all positives or all negatives)**, `resampling_before_clustering` and `semi_resampling` **alleviated this phenomenon**, and `resampling_after_clustering` was **essentially balanced**.

Predicted ratios against training ratios

Figure S116 puts the predicted ratios in context of the training ratios, elucidating a variety of trends:

- `no_resampling`: **positive trend between the training and the predicted ratio**, but since the training and the test ratio also positively correlated (figure S111), the latter could be the one driving the predicted ratio of positives.
- `resampling_after_clustering` had a **constant training ratio**, meaning that the predicted ratio was not explainable by differences in training ratios.

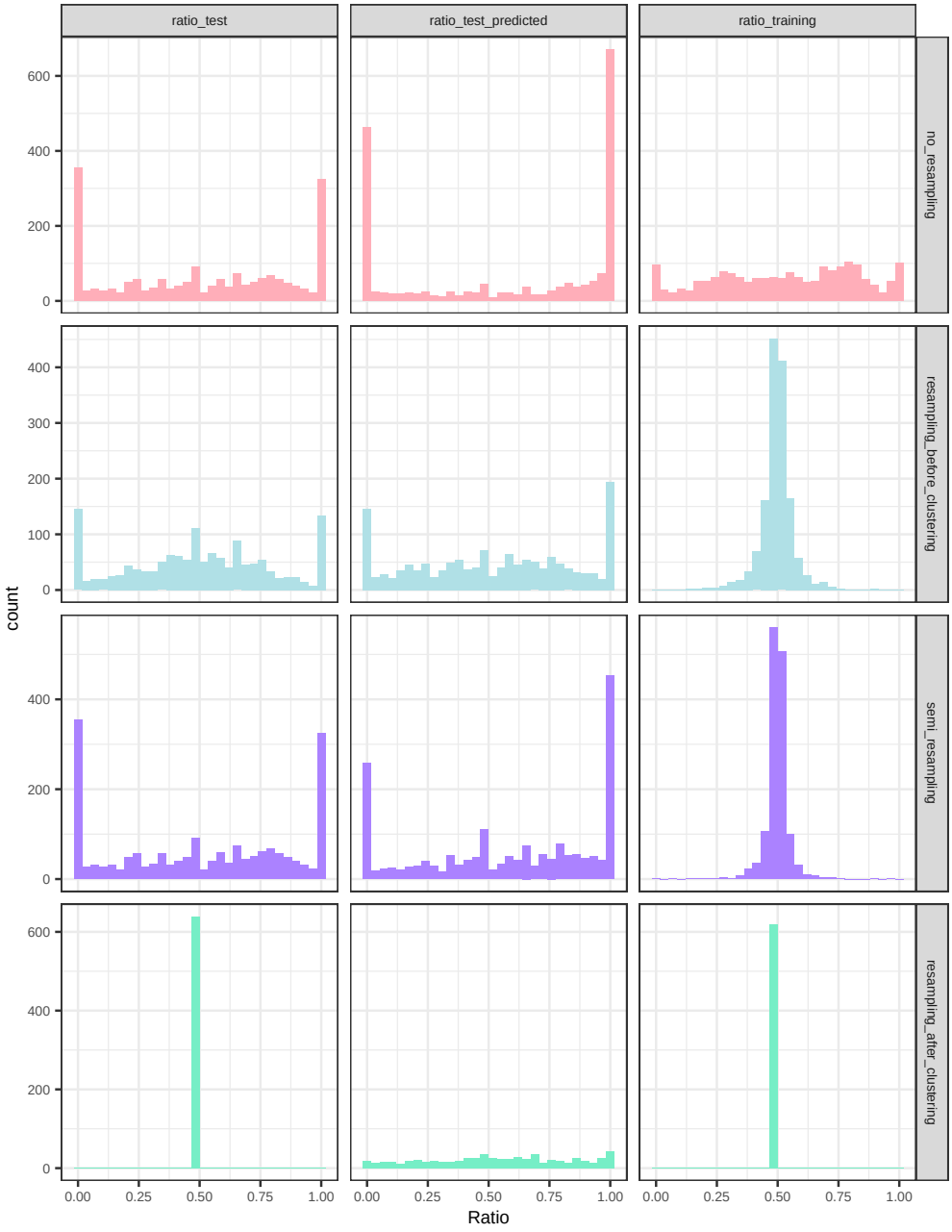

Figure S115: Distributions of the actives ratio in the training set and in the test set (both original and predicted by the deep learning model).

Table S89: Percentage of extreme cases, i.e. proteins with all actives or inactives.

strategy	RatioSet	all_inactives	all_actives	all_extremes
no_resampling	ratio_test	18.4	16.9	35.3
no_resampling	ratio_test_predicted	23.8	33.6	57.5
no_resampling	ratio_training	4.8	5.3	10.1
resampling_before_clustering	ratio_test	10.0	9.2	19.2
resampling_before_clustering	ratio_test_predicted	9.6	12.8	22.4
resampling_before_clustering	ratio_training	0.1	0.0	0.1
semi_resampling	ratio_test	18.4	16.9	35.3
semi_resampling	ratio_test_predicted	13.5	23.4	36.9
semi_resampling	ratio_training	0.1	0.0	0.1
resampling_after_clustering	ratio_test	0.0	0.0	0.0
resampling_after_clustering	ratio_test_predicted	2.7	4.5	7.2
resampling_after_clustering	ratio_training	0.0	0.0	0.0

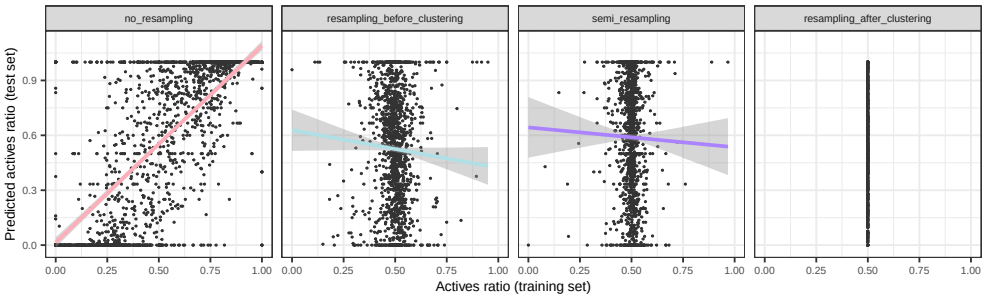

Figure S116: Predicted ratios, as a function of training ratios.

- `resampling_before_clustering` showed a **negative but not significant coefficient in PRs (p=0.073)**.
- `semi_resampling` showed **no apparent correlation** between the predicted ratio and the training ratio.

The significance of the linear correlation backs up all the claims above (table S90).

Table S90: Correlations between train and predicted test active ratios. 95% confidence intervals and p-values are shown.

strategy	cor	ci_lower	ci_upper	alternative	pvalue
no_resampling	0.732	0.711	0.753	two.sided	3.35e-312
resampling_before_clustering	-0.047	-0.098	0.004	two.sided	7.29e-02
semi_resampling	-0.017	-0.069	0.035	two.sided	5.20e-01
resampling_after_clustering	NA	NA	NA	two.sided	NA

Linear models

The predicted ratio of actives r_{pred} was modelled through the following quasibinomial generalized linear models, stratified by strategy:

$$r_{pred} \sim r_{training} + r_{test} + \log10(n_{int}) + \log10(n_{seq}) + k_{fold}$$

The main variables of interest are the actual ratios in the training $r_{training}$ and in test r_{test}, both numeric between 0 and 1. As additional covariates,

the number of interactions n_{int} and the sequence length n_{seq} (numerical) and the fold number k_{fold} (categorical) were also included.

IN SEMI_RESAMPLING OR RESAMPLING_BEFORE_CLUSTERING **Due to the lack of correlation between training and test ratios** (figure S111), the `semi_resampling` strategy is the **ideal scenario to disentangle their effects** on the predicted ratio of actives (see model in table S91). This additive model suggests:

- The **test ratio is driving the predicted proportions, rather than the training ratio**.
- `n_interactions`: **the term is not significant in PRs**.

Table S91 also shows the additive model for `resampling_before_clustering`. This strategy showed negative correlation between training and test ratios, also providing a reasonably good scenario to distinguish their effects.

- **This model confirms both conclusions from the model** in the `semi_resampling` strategy. **The coefficient for ratio test is similar and significant in both models; the coefficient of ratio training is non-significant in both models.**

IN NO_RESAMPLING The explanatory linear model under the `no_resampling` strategy (table S92) **suffers from the positive correlation between training and test ratios, which can be confounded**.

- Both `training_ratio` and `test_ratio` show a **positive effect** on the predicted fraction of actives.
- Although the **estimate is larger and more significant** for `training_ratio`, the **confounding effect of the predicted ratios deems this model inconclusive**. **For PRs the distribution of predicted ratios is not very skewed**.

Conclusions

- Data imbalance exists in all strategies but in `resampling_after_clustering`, where balance is enforced.
- The correlation between a protein's ratio in train and test is positive in `no_resampling`, negative in `resampling_before_clustering` and null in `semi_resampling` and `resampling_after_clustering`.
- The main factor driving the ratio of actives in the model predictions, per protein, is the actual ratio of positives in the test set. Their distributions are resemblant, and linear models confirm the association.

All of them apply to PRs as well, except for the negative correlation in resampling before clustering, which is non-significant in PRs.

Table S91: Linear models to describe the predicted active ratio for the semi_resampling and the resampling_before_clustering strategies. Significance and 95% confidence intervals are included.

	semi_resampling	resampling_before_clustering
	(1)	(2)
ratio_training	−0.193 (−1.563, 1.177)	0.872 (−0.109, 1.853)
	p = 0.783	p = 0.082
ratio_test	1.149 (0.931, 1.368)***	1.178 (0.928, 1.428)***
	p = 5.340e-24	p = 9.248e-20
log10(n_interactions)	−0.044 (−0.174, 0.086)	−7.026e-03 (−0.123, 0.109)
	p = 0.51	p = 0.905
log10(len_seq)	−0.451 (−0.804, −0.097)*	−0.163 (−0.5, 0.173)
	p = 0.013	p = 0.342
fold1	0.324 (0.014, 0.635)*	0.491 (0.19, 0.792)*
	p = 0.041	p = 1.418e-03
fold2	0.421 (0.115, 0.726)*	0.297 (−2.457e-03, 0.597)
	p = 6.999e-03	p = 0.052
fold3	0.857 (0.537, 1.177)***	0.914 (0.606, 1.221)***
	p = 1.839e-07	p = 6.974e-09
fold4	0.011 (−0.291, 0.312)	0.213 (−0.08, 0.505)
	p = 0.945	p = 0.155
fold5	−0.182 (−0.486, 0.123)	0.015 (−0.278, 0.309)
	p = 0.243	p = 0.92
fold6	−0.516 (−0.815, −0.216)**	−0.343 (−0.642, −0.045)*
	p = 7.708e-04	p = 0.024
fold7	−0.509 (−0.81, −0.208)**	0.164 (−0.136, 0.463)
	p = 9.439e-04	p = 0.285
fold8	0.2 (−0.102, 0.501)	0.086 (−0.208, 0.381)
	p = 0.195	p = 0.567
fold9	0.413 (0.092, 0.733)*	0.07 (−0.236, 0.376)
	p = 0.012	p = 0.655
Constant	1.12 (−0.08, 2.32)	−0.641 (−1.757, 0.475)
	p = 0.068	p = 0.26
Observations	1405	1452

Note: *p<0.05; **p<1.000e-03; ***p<1e-06

Table S92: Linear models to describe the predicted active ratio for the no_resampling strategy. Significance and 95% confidence intervals are included.

	no_resampling
ratio_training	6.124 (5.564, 6.684)***
	p = 3.380e-91
ratio_test	1.491 (1.131, 1.851)***
	p = 8.721e-16
log10(n_interactions)	−0.547 (−0.706, −0.388)***
	p = 2.010e-11
log10(len_seq)	−0.298 (−0.831, 0.236)
	p = 0.275
fold1	0.585 (0.111, 1.06)*
	p = 0.016
fold2	0.494 (0.039, 0.948)*
	p = 0.033
fold3	1.851 (1.363, 2.34)***
	p = 1.682e-13
fold4	0.522 (0.07, 0.974)*
	p = 0.024
fold5	−0.317 (−0.795, 0.161)
	p = 0.193
fold6	−0.367 (−0.831, 0.097)
	p = 0.121
fold7	0.486 (0.026, 0.946)*
	p = 0.038
fold8	0.383 (−0.071, 0.838)
	p = 0.099
fold9	0.363 (−0.113, 0.84)
	p = 0.135
Constant	−1.794 (−3.288, −0.301)*
	p = 0.019
Observations	1858
Note:	*p<0.05; **p<1.000e-03; ***p<1e-06

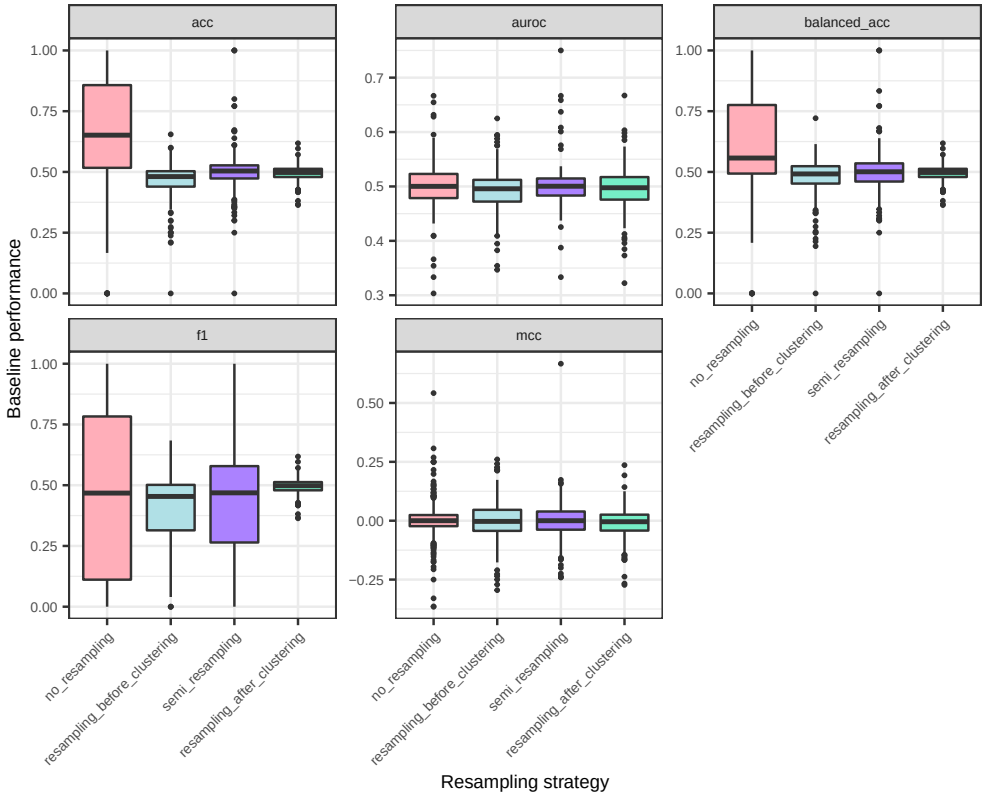

Figure S117: Performance metrics for imbalance-aware random baselines. Data points correspond to proteins, averaged over folds.

C.4.4 Description of baseline performance

Before evaluating the deep learning model, the performance metrics of the baselines were characterised, in order to pinpoint imbalance-sensitive and insensitive metrics. Metrics were called imbalance-sensitive if the imbalance-aware random baseline exhibited different performances between resampling strategies.

Descriptive plot

Figure S117 shows a fold-averaged picture of the metrics by protein. **Visual inspection suggested that accuracy, F1 and possibly balanced accuracy were affected by the data imbalance**. **accuracy is the most apparent case in PRs**, see the quartiles in table S93.

Table S93: Quartiles for the baseline F1-scores.

strategy	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
no_resampling	0.000	0.111	0.468	0.455	0.783	1.000
resampling_before_clustering	0.000	0.314	0.454	0.406	0.501	0.684
semi_resampling	0.000	0.264	0.469	0.424	0.579	1.000
resampling_after_clustering	0.364	0.479	0.498	0.493	0.513	0.618

Linear models

Formally, each performance metric was described with the following linear model:

$$\text{metric} \sim \text{strategy} + \log10(n_{int}) + \log10(n_{seq}) + k_{fold}$$

The response was the quantitative metric of interest (one model per metric), while `strategy` was categorical with the following possibilities: `no_resampling`, `resampling_after_clustering`, `resampling_before_clustering`, `semi_resampling`. Additional covariates included the number of interactions n_{int} and the sequence length n_{seq} (numerical) and the fold number k_{fold} (categorical). The `strategy` variable was tested with a type 3 ANOVA, being **significant** with $p < 0.05$ for `acc`, `f1` and `balanced_acc` (table S94).

Table S94: ANOVA p-values for including the resampling strategy as a regressor. Significant p-values imply that differences exist between resampling strategies.

strategy	variable	Sum Sq	Df	F value	Pr(>F)
acc	strategy	26.3568423	3	2.18e+02	3.44e-133
auroc	strategy	0.0655018	3	6.69e-01	5.71e-01
balanced_acc	strategy	7.6709749	3	5.67e+01	4.59e-36
f1	strategy	5.5240885	3	2.42e+01	1.56e-15
mcc	strategy	0.0668456	3	4.14e-01	7.43e-01

Based on this, metrics were divided in two types:

- Those where the baseline was different between strategies, i.e. imbalance-sensitive: `acc`, `f1` and `balanced_acc`. Therefore, before comparing strategies, the baseline performance needed to be accounted for.
- Those where the baseline was constant, i.e. imbalance-insensitive: `auroc`, `mcc`. Here we could compare strategies directly.

All applies to PRs as well.

C.4.5 Description of deep learning model performance

An overview of fold-averaged performances is displayed in figure S118, where strategies are paired with their baselines. This illustrates the **issue of direct strategy comparison** with imbalance-sensitive metrics, which was especially visible for the F1-score. Some metrics are undefined in edge cases (e.g. AUROC when only actives or only inactives are available); table S95 summarizes the number of proteins, added over folds, whose metrics were computable.

Absolute, baseline-naive performance

Anologous to the baseline performance models, absolute metric models (not accounting for baselines) were fitted:

Table S95: Number of computable performance measures. AUROC was undefined for proteins with all actives or unactives in the test set, hence its lower counts.

strategy	acc	auroc	f1	balanced_acc	mcc
no_resampling	1896	1227	1896	1896	1896
resampling_before_clustering	1473	1199	1473	1473	1473
semi_resampling	1896	1227	1896	1896	1896
resampling_after_clustering	639	639	639	639	639

$$\text{metric} \sim \text{strategy} + \log10(n_{int}) + \log10(n_{seq}) + k_{fold}$$

The `strategy` covariate was **always significant** in a type 3 ANOVA (table S96). The models, summarized in S97, showed **different behaviour in imbalance-sensitive and insensitive metrics**. Pairwise comparisons of the strategy coefficients using Tukey's method would point to **two different pictures** (figure S119), further confirmed when prioritizing the strategies according to their expected performance through the linear models (table S98 and figure S120):

- **Accuracy and balanced accuracy** suggested that `no_resampling` was the **best strategy**, but this was confounded by the fact that it also held the **highest baselines**.
- **In PRs, MCC suggested resampling before clustering and semi resampling as the best strategies**.
- **In PRs, F1 and AUROC also suggested the no resampling strategy as the best one, but its baselines were not higher**.

Table S96: ANOVA p-values for including the resampling strategy as a regressor in the performance models.

strategy	variable	Sum Sq	Df	F value	Pr(>F)
acc	strategy	17.362539	3	7.55e+01	6.14e-48
auroc	strategy	1.090834	3	6.37e+00	2.65e-04
f1	strategy	2.170764	3	5.85e+00	5.54e-04
balanced_acc	strategy	1.524648	3	7.34e+00	6.61e-05
mcc	strategy	1.027279	3	5.16e+00	1.46e-03

Table S97: Linear models to describe each performance metric. Standard deviations in parentheses.

	acc	auroc	f1	balanced_acc	mcc
	(1)	(2)	(3)	(4)	(5)
strategyresampling_before_clustering	−0.109***	−0.011	−0.045**	−0.023*	0.028*
	(9.727e-03)	(9.718e-03)	(0.012)	(9.248e-03)	(9.053e-03)
	p = 9.079e-29	p = 0.262	p = 2.456e-04	p = 0.013	p = 2.249e-03
strategysemi_resampling	−0.095***	−5.248e-03	−0.012	−0.026*	0.028**
	(8.990e-03)	(9.647e-03)	(0.011)	(8.548e-03)	(8.367e-03)
	p = 5.466e-26	p = 0.586	p = 0.311	p = 1.999e-03	p = 7.070e-04
strategyresampling_after_clustering	−0.164***	−0.05**	−0.048*	−0.054**	6.680e-03
	(0.013)	(0.012)	(0.017)	(0.012)	(0.012)
	p = 1.868e-35	p = 2.966e-05	p = 4.330e-03	p = 1.362e-05	p = 0.585
log10(n_interactions)	0.04***	0.068***	0.121***	6.244e-03	0.082***
	(5.085e-03)	(6.200e-03)	(6.463e-03)	(4.834e-03)	(4.732e-03)
	p = 9.245e-15	p = 1.322e-27	p = 1.553e-75	p = 0.197	p = 2.538e-66
log10(len_seq)	0.059*	0.011	−0.044	0.031	−0.017
	(0.018)	(0.019)	(0.023)	(0.017)	(0.017)
	p = 1.028e-03	p = 0.55	p = 0.055	p = 0.07	p = 0.31
fold1	0.044*	−4.954e-03	0.035	0.031	−0.019
	(0.016)	(0.017)	(0.021)	(0.016)	(0.015)
	p = 7.992e-03	p = 0.769	p = 0.09	p = 0.05	p = 0.206
fold2	−0.035*	−0.012	0.013	−0.035*	−7.381e-03
	(0.016)	(0.016)	(0.02)	(0.015)	(0.015)
	p = 0.029	p = 0.463	p = 0.513	p = 0.022	p = 0.618
fold3	−0.036*	7.250e-03	0.057*	−0.036*	2.073e-03
	(0.016)	(0.016)	(0.02)	(0.015)	(0.015)
	p = 0.024	p = 0.654	p = 5.524e-03	p = 0.02	p = 0.89
fold4	1.550e-03	8.425e-03	0.016	−0.015	3.890e-03
	(0.016)	(0.016)	(0.02)	(0.015)	(0.015)
	p = 0.922	p = 0.593	p = 0.417	p = 0.314	p = 0.792
fold5	0.024	0.037*	9.275e-03	0.013	0.037*
	(0.016)	(0.017)	(0.021)	(0.016)	(0.015)
	p = 0.148	p = 0.024	p = 0.658	p = 0.408	p = 0.016
fold6	0.039*	0.022	−0.017	0.03	0.012
	(0.016)	(0.016)	(0.021)	(0.015)	(0.015)
	p = 0.017	p = 0.183	p = 0.404	p = 0.053	p = 0.438
fold7	1.006e-03	0.021	7.738e-03	−7.655e-04	0.015
	(0.016)	(0.016)	(0.021)	(0.015)	(0.015)
	p = 0.95	p = 0.193	p = 0.706	p = 0.96	p = 0.316
fold8	0.025	0.015	0.043*	0.01	0.011
	(0.016)	(0.016)	(0.02)	(0.015)	(0.015)
	p = 0.122	p = 0.354	p = 0.035	p = 0.501	p = 0.471
fold9	0.04*	0.039*	0.058*	0.023	0.034*
	(0.017)	(0.017)	(0.021)	(0.016)	(0.015)
	p = 0.016	p = 0.019	p = 6.344e-03	p = 0.137	p = 0.027
Constant	0.438***	0.403***	0.331***	0.505***	−0.082
	(0.051)	(0.054)	(0.065)	(0.048)	(0.047)
	p = 1.107e-17	p = 6.987e-14	p = 3.247e-07	p = 3.037e-25	p = 0.085
Observations	5904	4292	5904	5904	5904
R^2	0.052	0.034	0.061	0.012	0.06
Adjusted R^2	0.05	0.03	0.059	9.687e-03	0.058

Note: *p<0.05; **p<1.000e-03; ***p<1e-06

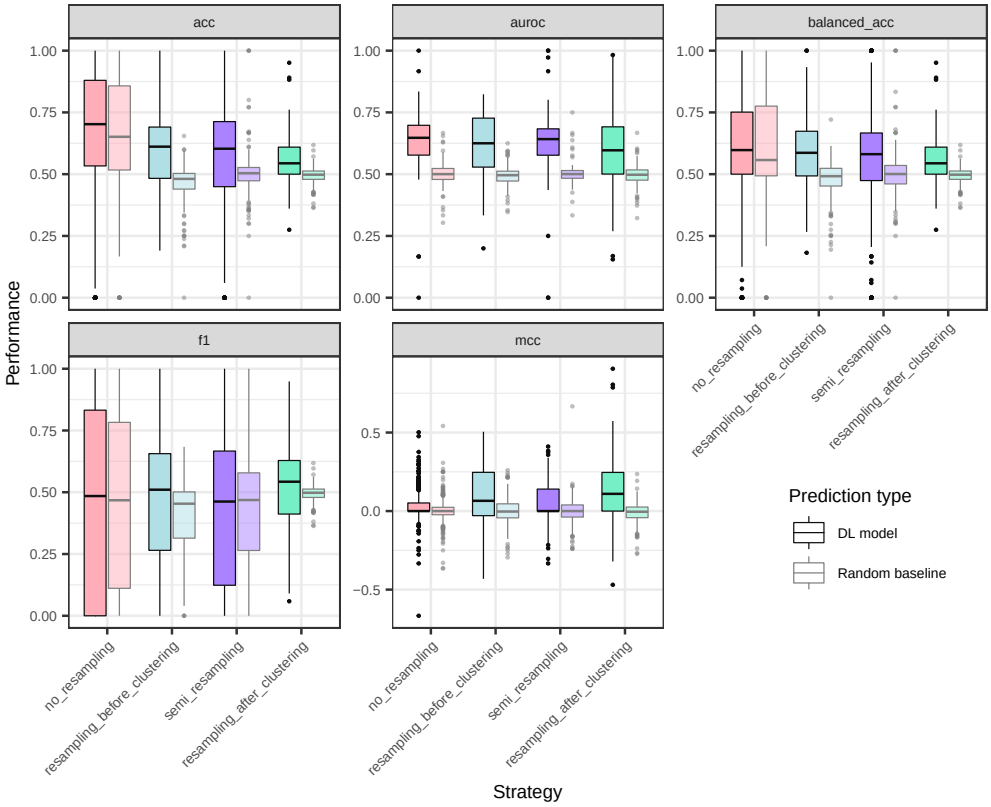

Figure S118: Performance metrics for balancing strategies and their corresponding imbalance-aware random baselines. Data points correspond to proteins, averaged over folds.

Table S98: Expected absolute performances, by metric and strategy, with 95% confidence intervals.

metric	strategy	emmean	SE	lower.CL	upper.CL
acc	no_resampling	0.720	7.179e-03	0.706	0.734
acc	resampling_before_clustering	0.611	7.476e-03	0.596	0.625
acc	semi_resampling	0.624	7.179e-03	0.610	0.638
acc	resampling_after_clustering	0.555	1.099e-02	0.534	0.577
auroc	no_resampling	0.634	7.122e-03	0.621	0.648
auroc	resampling_before_clustering	0.624	7.105e-03	0.610	0.637
auroc	semi_resampling	0.629	7.122e-03	0.615	0.643
auroc	resampling_after_clustering	0.585	9.500e-03	0.566	0.604
f1	no_resampling	0.566	9.125e-03	0.548	0.584
f1	resampling_before_clustering	0.520	9.503e-03	0.502	0.539
f1	semi_resampling	0.554	9.125e-03	0.536	0.572
f1	resampling_after_clustering	0.518	1.396e-02	0.491	0.546
balanced_acc	no_resampling	0.610	6.826e-03	0.597	0.623
balanced_acc	resampling_before_clustering	0.587	7.109e-03	0.573	0.601
balanced_acc	semi_resampling	0.584	6.826e-03	0.570	0.597
balanced_acc	resampling_after_clustering	0.556	1.045e-02	0.535	0.576
mcc	no_resampling	0.108	6.682e-03	0.095	0.121
mcc	resampling_before_clustering	0.136	6.959e-03	0.122	0.150
mcc	semi_resampling	0.137	6.682e-03	0.123	0.150
mcc	resampling_after_clustering	0.115	1.023e-02	0.095	0.135

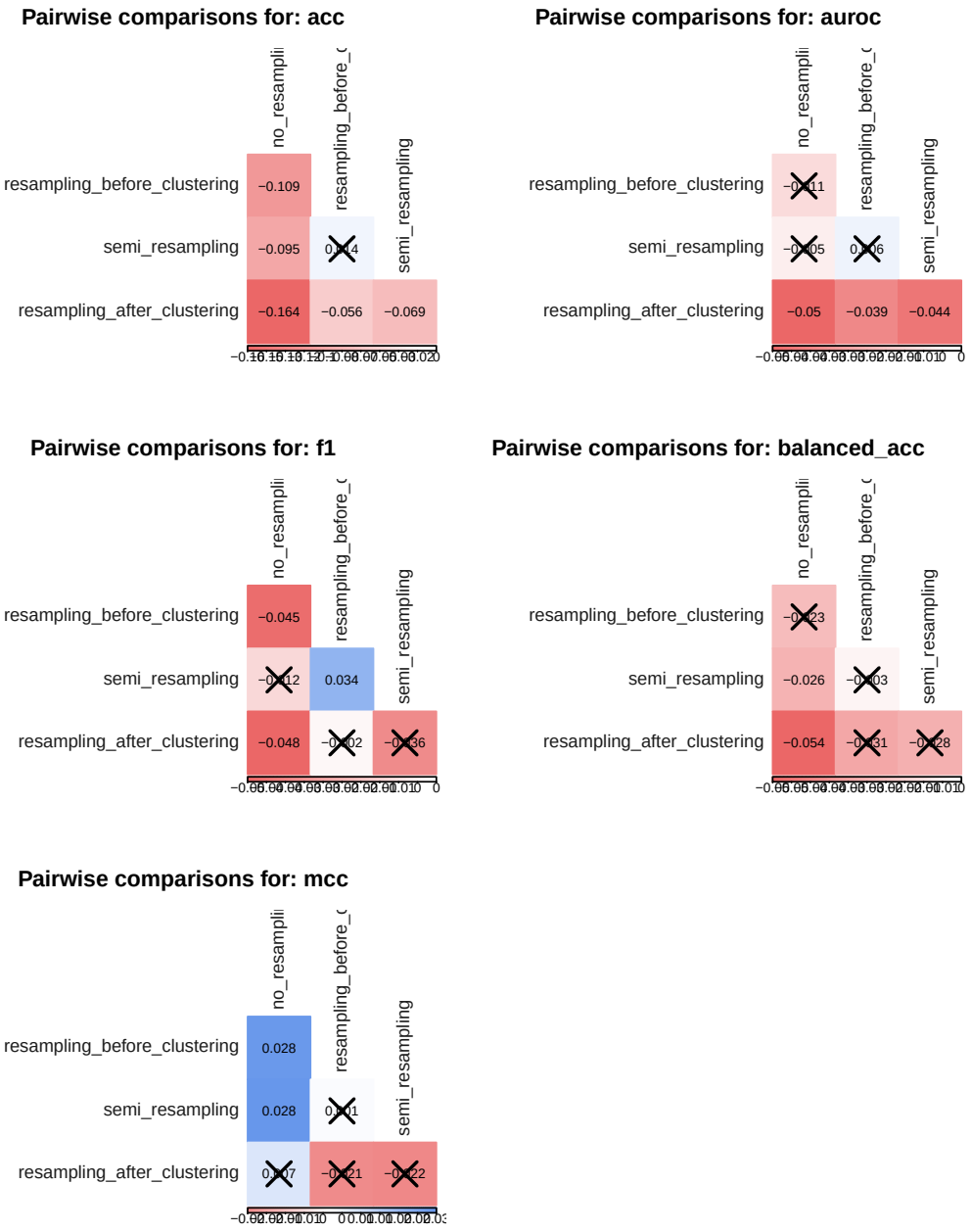

Figure S119: Pairwise comparison of strategy performance using Tukey method.

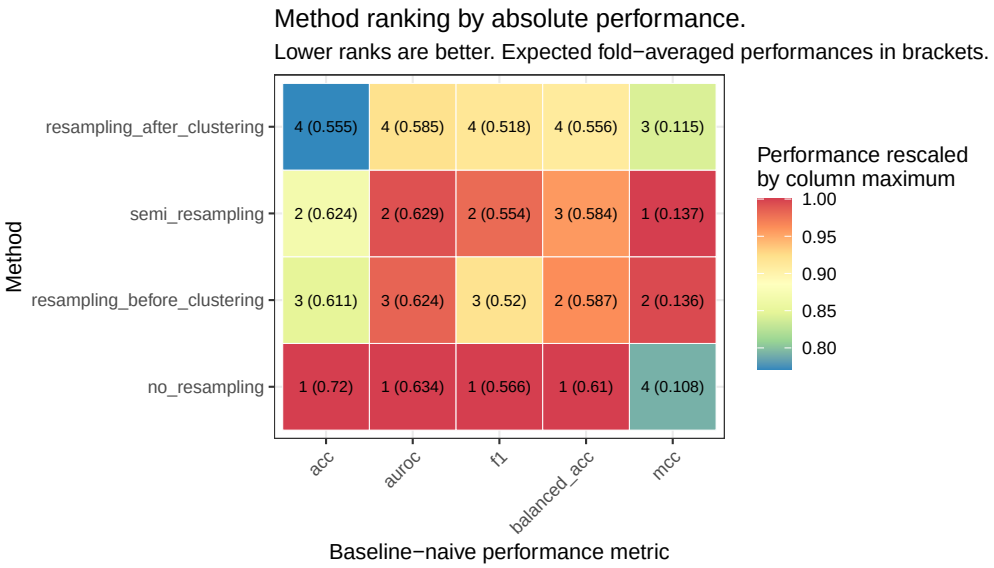

Figure S120: Method ranking according to the linear model predicted performances for each metric. Baseline metrics were ignored.

Baseline-adjusted performance

To address the pitfalls of the direct comparison of metrics whose baselines may differ, baseline-adjusted performance metrics were defined and modelled analogously. Specifically:

$$\text{adj\_metric} = \text{metric} - \text{baseline}$$

A descriptive plot of the the adjusted metrics (figure S121) pointed to a **scenario different than that of unadjusted ones** (figure S118).

Adjusted performance metrics were described with the following linear model:

$$\text{adj\_metric} \sim \text{strategy} + \log10(n_{int}) + \log10(n_{seq}) + k_{fold}$$

Note that while all metrics but `mcc` were non-negative, the adjusted metrics could show negative values when the performance of the DL model was lower than that of the baseline.

Again, `strategy` covariate was **always significant** in a type 3 ANOVA (table S99). **Baseline adjustment brought a uniform behaviour across the models, except for AUROC still suggesting no resampling as the best strategy in PRs** (table S100), further confirmed in pairwise coefficient comparison (Tukey's method, figure S122) and in their expected performance (table S101 and figure S123):

- **In PRs, resampling before clustering was the best performer, closely followed by semi resampling, then no resampling, and resampling after clustering**.
- **In PRs the general picture of resampling before clustering and semi resampling being the best was already present in unadjusted MCC, and is mostly kept in ajdusted metrics**.

Conclusions drawn from the baseline-adjusted performance analysis:

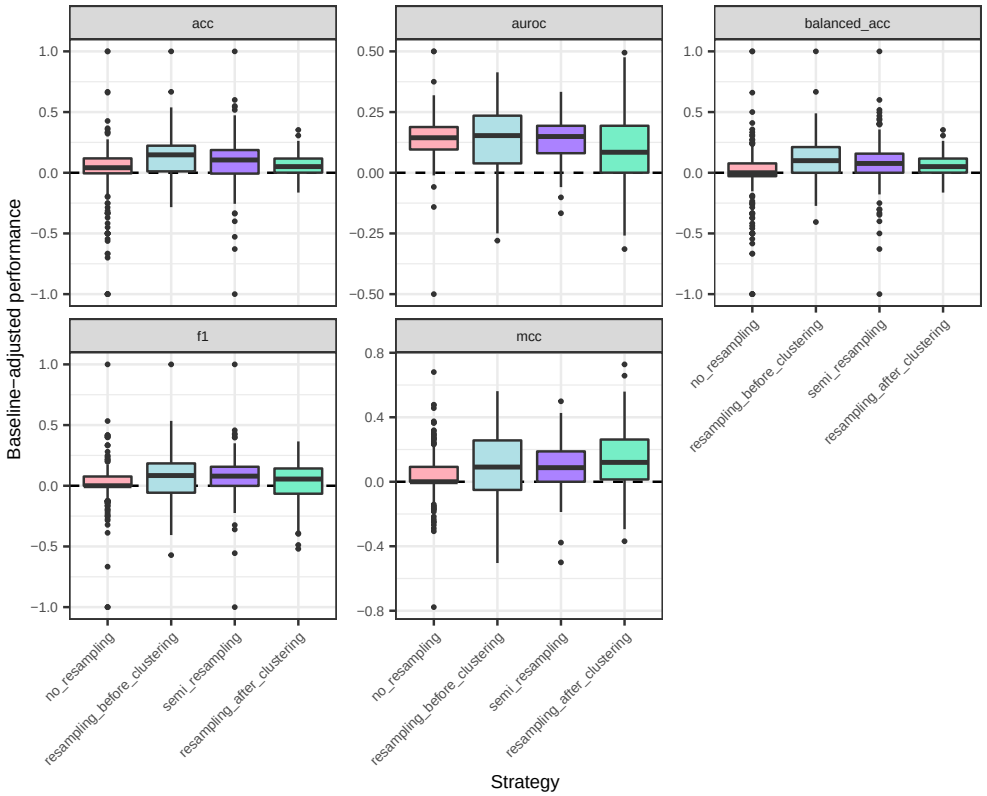

Figure S121: Baseline-adjusted performance metrics for balancing strategies. Data points correspond to proteins, averaged over folds.

Table S99: ANOVA p-values for including the resampling strategy as a regressor in the adjusted performance models.

strategy	variable	Sum Sq	Df	F value	Pr(>F)
acc	strategy	4.591602	3	1.82e+01	1.02e-11
auroc	strategy	1.116042	3	4.09e+00	6.55e-03
f1	strategy	2.165202	3	1.01e+01	1.30e-06
balanced_acc	strategy	4.523000	3	1.80e+01	1.30e-11
mcc	strategy	1.024287	3	2.82e+00	3.77e-02

- **In PRs, resampling before clustering performed similarly to semi resampling, and better than no resampling and resampling after clustering. Therefore, augmenting the test set was not the largest performance driver**.
- `semi_resampling` outperformed `no_resampling` in **two out of five metrics, and non-significant in the remaining three** (Tukey's method, $p < 0.05$, figure S122), which **supports data augmentation usefulness** even if the data balance in the test set differs from that of the training set. This was **consistent with the observation that the predicted proportion of positives of the PCM model was mainly driven by the actual data balance in the test set, rather than that of the training set. Combined with the healthier distributions of predicted active ratios** of `semi_resampling` against `no_resampling`, this made a **case in favour of the former**.

Table S100: Linear models to describe each adjusted performance metric. Standard deviations in parentheses.

	acc	auroc	f1	balanced_acc	mcc
	(1)	(2)	(3)	(4)	(5)
strategyresampling_before_clustering	0.056***	−6.020e-03	0.027*	0.069***	0.033*
	(0.01)	(0.012)	(9.460e-03)	(0.01)	(0.012)
	p = 3.898e-08	p = 0.624	p = 4.670e-03	p = 1.650e-11	p = 8.010e-03
strategysemi_resampling	0.027*	−0.011	0.041**	0.045**	0.023
	(0.01)	(0.013)	(9.629e-03)	(0.01)	(0.013)
	p = 0.011	p = 0.376	p = 1.816e-05	p = 1.482e-05	p = 0.062
strategyresampling_after_clustering	−0.035*	−0.051**	−0.016	6.630e-03	4.841e-03
	(0.014)	(0.015)	(0.013)	(0.014)	(0.017)
	p = 0.014	p = 7.690e-04	p = 0.23	p = 0.639	p = 0.776
log10(n_interactions)	0.061***	0.073***	0.032***	0.044***	0.093***
	(6.353e-03)	(8.505e-03)	(5.859e-03)	(6.335e-03)	(7.619e-03)
	p = 1.528e-21	p = 8.457e-18	p = 6.161e-08	p = 2.983e-12	p = 7.318e-34
log10(len_seq)	0.051*	3.223e-03	−0.037*	0.021	−0.025
	(0.02)	(0.024)	(0.018)	(0.02)	(0.024)
	p = 0.011	p = 0.894	p = 0.045	p = 0.281	p = 0.297
fold1	0.036*	0.02	0.041*	0.018	−2.052e-03
	(0.018)	(0.022)	(0.017)	(0.018)	(0.022)
	p = 0.045	p = 0.355	p = 0.013	p = 0.316	p = 0.924
fold2	−0.032	0.014	0.026	−0.025	0.015
	(0.018)	(0.02)	(0.016)	(0.017)	(0.021)
	p = 0.071	p = 0.506	p = 0.105	p = 0.145	p = 0.474
fold3	−0.015	0.042*	0.093***	−0.015	0.032
	(0.018)	(0.021)	(0.016)	(0.018)	(0.021)
	p = 0.381	p = 0.042	p = 1.039e-08	p = 0.382	p = 0.129
fold4	0.028	0.028	0.04*	0.013	0.022
	(0.017)	(0.02)	(0.016)	(0.017)	(0.021)
	p = 0.104	p = 0.163	p = 0.014	p = 0.456	p = 0.285
fold5	0.033	0.046*	0.012	0.023	0.043*
	(0.018)	(0.021)	(0.017)	(0.018)	(0.022)
	p = 0.069	p = 0.031	p = 0.474	p = 0.192	p = 0.046
fold6	0.063**	0.049*	9.638e-03	0.061**	0.043*
	(0.018)	(0.021)	(0.016)	(0.018)	(0.021)
	p = 4.240e-04	p = 0.019	p = 0.556	p = 6.183e-04	p = 0.043
fold7	8.429e-04	0.031	0.015	−6.795e-04	0.018
	(0.018)	(0.021)	(0.016)	(0.018)	(0.021)
	p = 0.962	p = 0.143	p = 0.353	p = 0.969	p = 0.405
fold8	0.028	0.028	0.039*	0.02	0.04
	(0.017)	(0.02)	(0.016)	(0.017)	(0.021)
	p = 0.111	p = 0.168	p = 0.017	p = 0.259	p = 0.057
fold9	0.031	0.044*	0.041*	0.014	0.037
	(0.018)	(0.021)	(0.017)	(0.018)	(0.022)
	p = 0.09	p = 0.039	p = 0.014	p = 0.453	p = 0.085
Constant	−0.235**	−0.106	0.025	−0.144*	−0.1
	(0.057)	(0.07)	(0.052)	(0.057)	(0.068)
	p = 3.387e-05	p = 0.127	p = 0.628	p = 0.011	p = 0.14
Observations	5355	4134	5355	5355	5355
R^2	0.038	0.021	0.021	0.028	0.035
Adjusted R^2	0.035	0.018	0.018	0.026	0.033

Note: *p<0.05; **p<1.000e-03; ***p<1e-06

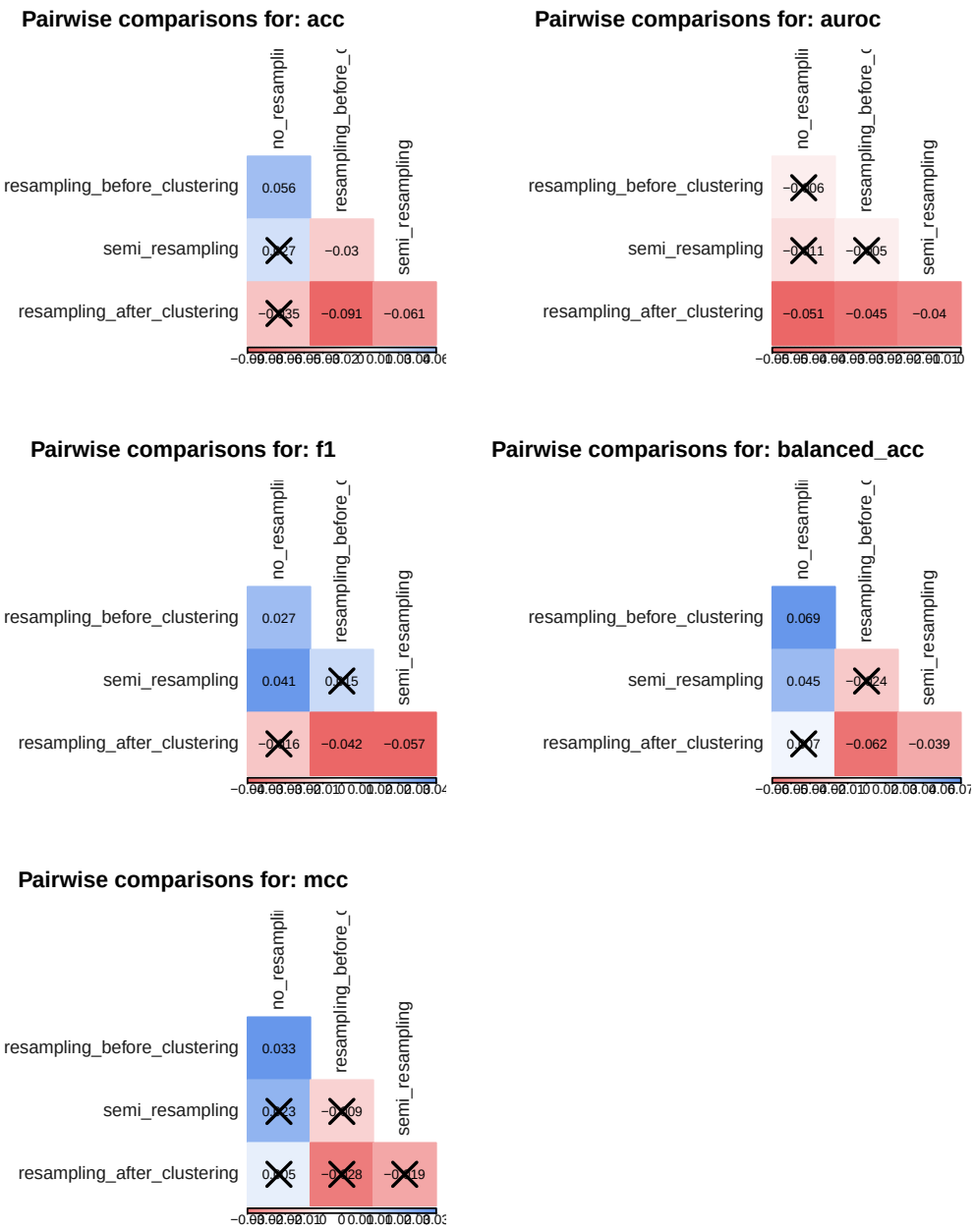

Figure S122: Pairwise comparison of strategy adjusted performance using Tukey method.

Table S101: Expected adjusted performances, by metric and strategy, with 95% confidence intervals.

metric	strategy	emmean	SE	lower.CL	upper.CL
acc	no_resampling	0.092	7.925e-03	0.077	0.108
acc	resampling_before_clustering	0.149	8.018e-03	0.133	0.165
acc	semi_resampling	0.119	8.091e-03	0.103	0.135
acc	resampling_after_clustering	0.057	1.171e-02	0.035	0.080
auroc	no_resampling	0.138	9.147e-03	0.121	0.156
auroc	resampling_before_clustering	0.132	9.093e-03	0.115	0.150
auroc	semi_resampling	0.127	9.347e-03	0.109	0.146
auroc	resampling_after_clustering	0.087	1.218e-02	0.063	0.111
f1	no_resampling	0.045	7.308e-03	0.030	0.059
f1	resampling_before_clustering	0.071	7.394e-03	0.057	0.086
f1	semi_resampling	0.086	7.461e-03	0.071	0.100
f1	resampling_after_clustering	0.029	1.080e-02	0.008	0.050
balanced_acc	no_resampling	0.050	7.903e-03	0.035	0.066
balanced_acc	resampling_before_clustering	0.119	7.996e-03	0.103	0.135
balanced_acc	semi_resampling	0.095	8.068e-03	0.079	0.111
balanced_acc	resampling_after_clustering	0.057	1.168e-02	0.034	0.080
mcc	no_resampling	0.117	9.504e-03	0.098	0.135
mcc	resampling_before_clustering	0.149	9.616e-03	0.130	0.168
mcc	semi_resampling	0.140	9.703e-03	0.121	0.159
mcc	resampling_after_clustering	0.121	1.404e-02	0.094	0.149

- In **five** out of five metrics, **proteins with more interactions were better predicted** (table S100).

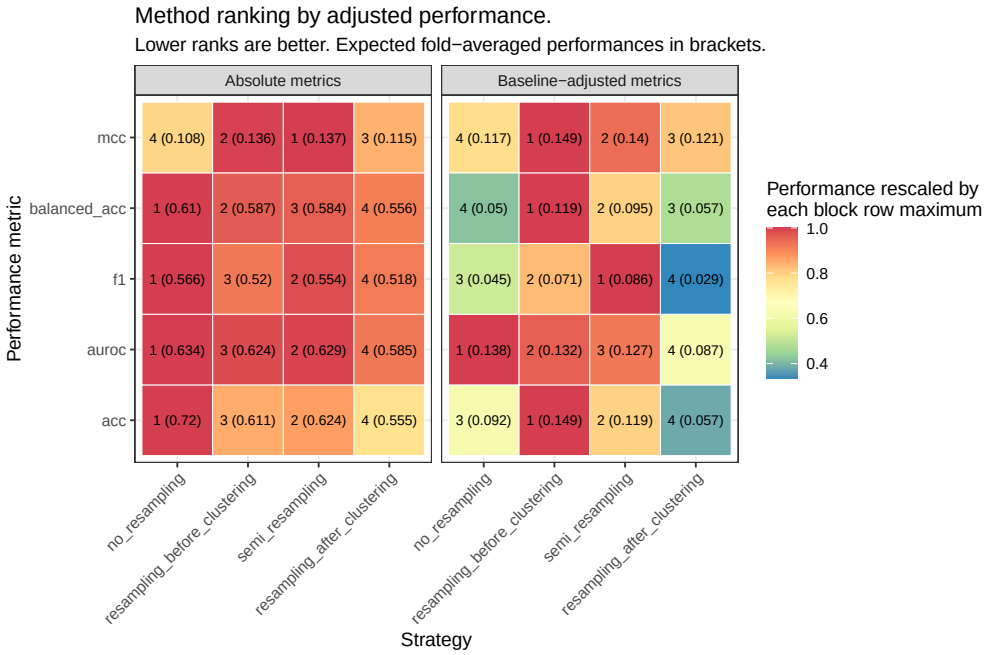

Figure S123: Method ranking according to the linear model predicted adjusted performances for each metric. Baseline metrics were taken into account in the adjustment. For a direct comparison, the same ranking using the absolute metrics was kept side by side.

c.4.6 Reproducibility

- R version 3.6.3 (2020-02-29), `x86_64-pc-linux-gnu`
- Locale: `LC_CTYPE=en_US.UTF-8`, `LC_NUMERIC=C`, `LC_TIME=es_ES.UTF-8`, `LC_COLLATE=en_US.UTF-8`, `LC_MONETARY=es_ES.UTF-8`, `LC_MESSAGES=en_US.UTF-8`, `LC_PAPER=es_ES.UTF-8`, `LC_NAME=C`, `LC_ADDRESS=C`, `LC_TELEPHONE=C`, `LC_MEASUREMENT=es_ES.UTF-8`, `LC_IDENTIFICATION=C`
- Running under: `Ubuntu 16.04.7 LTS`
- Matrix products: default
- BLAS: `/usr/lib/atlas-base/atlas/libblas.so.3.0`
- LAPACK: `/usr/lib/atlas-base/atlas/liblapack.so.3.0`
- Base packages: base, datasets, graphics, grDevices, methods, stats, utils
- Other packages: corrplot 0.84, dplyr 0.8.3, forcats 0.5.0, ggplot2 3.3.0, gsubfn 0.7, kableExtra 1.3.1, magrittr 1.5, proto 1.0.0, purrr 0.3.3, readr 1.3.1, rmarkdown 1.12, stargazer 5.2.2, stringr 1.4.0, tibble 2.1.3, tidyr 1.0.0, tidyverse 1.3.0
- Loaded via a namespace (and not attached): abind 1.4-5, assertthat 0.2.1, backports 1.1.4, bookdown 0.12, broom 0.5.3, car 3.0-10, carData 3.0-4, cellranger 1.1.0, cli 1.1.0, codetools 0.2-16, colorspace 1.4-1, compiler 3.6.3, crayon 1.3.4, curl 3.3, data.table 1.12.2,

DBI 1.0.0, dbplyr 1.4.2, digest 0.6.18, emmeans 1.5.3, estimability 1.3, evaluate 0.13, foreign 0.8-76, fs 1.3.1, generics 0.0.2, glue 1.3.1, grid 3.6.3, gtable 0.3.0, haven 2.2.0, highr 0.8, hms 0.5.3, htmltools 0.3.6, httr 1.4.1, jsonlite 1.6, knitr 1.22, labeling 0.3, lattice 0.20-41, lifecycle 0.1.0, lubridate 1.7.4, MASS 7.3-53, Matrix 1.2-18, mgcv 1.8-33, modelr 0.1.6, multcomp 1.4-15, munsell 0.5.0, mvtnorm 1.0-11, nlme 3.1-149, openxlsx 4.1.0.1, pillar 1.4.3, pkgconfig 2.0.2, plyr 1.8.4, R6 2.4.0, RColorBrewer 1.1-2, Rcpp 1.0.5, readxl 1.3.1, reprex 0.3.0, reshape2 1.4.3, rio 0.5.16, rlang 0.4.5, rstudioapi 0.10, rvest 0.3.5, sandwich 2.5-1, scales 1.0.0, splines 3.6.3, stringi 1.4.3, survival 3.2-7, tcltk 3.6.3, TH.data 1.0-10, tidyselect 0.2.5, tools 3.6.3, vctrs 0.2.4, viridisLite 0.3.0, webshot 0.5.2, withr 2.1.2, xfun 0.6, xml2 1.2.5, xtable 1.8-4, yaml 2.2.0, zip 2.0.4, zoo 1.8-6

BIBLIOGRAPHY

Abraham, Mark James, Teemu Murtola, Roland Schulz, Szilárd Páll, Jeremy C Smith, Berk Hess, and Erik Lindahl

2015 "GROMACS: High Performance Molecular Simulations through Multi-Level Parallelism from Laptops to Supercomputers", *SoftwareX*, 1, pp. 19-25.

Agathocleous, Michalis, Georgia Christodoulou, Vasilis Promponas, Chris Christodoulou, Vassilis Vassiliades, and Antonis Antoniou

2010 "Protein Secondary Structure Prediction with Bidirectional Recurrent Neural Nets: Can Weight Updating for Each Residue Enhance Performance?", in *IFIP Int. Conf. Art. Int. Applic. Innov.* Springer, pp. 128-137.

Ahmad, Shandar and Akinori Sarai

2005 "PSSM-based Prediction of DNA Binding Sites in Proteins", *BMC Bioinf.* 6, pp. 1-6.

Alipanahi, Babak, Andrew Delong, Matthew T Weirauch, and Brendan J Frey

2015 "Predicting the Sequence Specificities of DNA- and RNA-binding Proteins by Deep Learning", *Nat. Biotechnol.* 33, 8, pp. 831-838.

Aliper, Alexander, Sergey Plis, Artem Artemov, Alvaro Ulloa, Polina Mamoshina, and Alex Zhavoronkov

2016 "Deep Learning Applications for Predicting Pharmacological Properties of Drugs and Drug Repurposing Using Transcriptomic Data", *Mol. Pharm.* 13, 7, pp. 2524-2530.

Allen, William J., Trent E. Balius, Sudipto Mukherjee, Scott R. Brozell, Demetri T. Moustakas, P. Therese Lang, David A. Case, Irwin D. Kuntz, and Robert C. Rizzo

2015 "DOCK 6: Impact of New Features and Current Docking Performance", *J. Comput. Chem.* 36, 15 (June 2015), pp. 1132-1156.

Almagro Armenteros, Jose Juan, Casper Kaae Sönderby, Sören Kaae Sönderby, Henrik Nielsen, and Ole Winther

2017 "DeepLoc: Prediction of Protein Subcellular Localization Using Deep Learning." *Bioinformatics*, 33, 21 (Nov. 2017), pp. 3387-3395.

Alonso, Hernán, Andrey A. Bliznyuk, and Jill E. Gready

2006 "Combining Docking and Molecular Dynamic Simulations in Drug Design", *Med. Res. Rev.* 26, 5 (Sept. 2006), pp. 531-568.

Altae-Tran, Han, Bharath Ramsundar, Aneesh S Pappu, and Vijay Pande

2017 "Low Data Drug Discovery with One-shot Learning", *ACS Central Sci.* 3, 4 (Apr. 2017), pp. 283-293.

And, Thorsten Naumann and Hans Matter

2002 "Structural Classification of Protein Kinases Using 3D Molecular Interaction Field Analysis of Their Ligand Binding Sites: Target Family Landscapes", *J. Med. Chem.* 12, 45, pp. 2366-2378.

Andersson, Claes R, Mats G Gustafsson, and Helena Strömbergsson

2011 "Quantitative Chemogenomics: Machine-Learning Models of Protein-Ligand Interaction." *Curr. Top. Med. Chem.* 11, 15, pp. 1978-1993.

Angermueller, Christof, Tanel Pärnamaa, Leopold Parts, and Oliver Stegle

2016 "Deep learning for Computational Biology", *Mol. Syst. Biol.* 12, 7, p. 878.

Asgari, Ehsaneddin, Mohammad R. K. Mofrad, WF Bluhm, CH Christie, D Dimitropoulos, and S Dutta

2015 "Continuous Distributed Representation of Biological Sequences for Deep Proteomics and Genomics", *PLOS ONE*, 10, 11 (Nov. 2015), e0141287.

Asgari, Ehsaneddin and Mohammad R.K. Mofrad

2019 *Deep Genomics and Proteomics: Language Model-Based Embedding of Biological Sequences and Their Applications in Bioinformatics*, Elsevier Inc., pp. 167-181, ISBN: 9780128095560.

Avsec, Žiga, Melanie Weilert, Avanti Shrikumar, Sabrina Krueger, and Amr Alexandari

2020 *Deep Learning at Base-Resolution Reveals Cis-regulatory Motif Syntax*.

Bahdanau, Dzmitry, Kyunghyun Cho, and Yoshua Bengio

2014 "Neural Machine Translation by Jointly Learning to Align and Translate", *arXiv*, 1409.0473 (Sept. 2014).

Bai, Shaojie, J Zico Kolter, and Vladlen Koltun

2018 "An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling", *arXiv*, 1803.01271.

Baldi, Pierre

2012 "Autoencoders, Unsupervised Learning, and Deep Architectures", *Proc. ICML Workshop Unsup. Transf. Learn.* 27, pp. 37-50.

Baldi, Pierre, Sören Brunak, Paolo Frasconi, Giovanni Soda, and Gianluca Pollastri

1999 "Exploiting the Past and the Future in Protein Secondary Structure Prediction", *Bioinformatics*, 15, 11 (Nov. 1999), pp. 937-46, ISSN: 1367-4803.

Baldi, Pierre and Gianluca Pollastri

2003 "The Principled Design of Large-Scale Recursive Neural Network Architectures–DAG-RNNs and the Protein Structure Prediction Problem", *J. Mach. Learn. Res.* 4, pp. 575-602.

Baldi, Pierre, Gianluca Pollastri, Claus A Andersen, and Sören Brunak

2000 "Matching Protein Beta-Sheet Partners by Feedforward and Recurrent Neural Networks.", in *Proc. Int. Conf. Int. Sys. Mol. Biol.* Vol. 8, pp. 25-36.

Bau, David, Bolei Zhou, Aditya Khosla, Aude Oliva, and Antonio Torralba

2017 "Network Dissection : Quantifying Interpretability of Deep Visual Representations", *arXiv*, 1704.05796v1.

Bemis, Guy W. and Mark A. Murcko

1996 "The Properties of Known Drugs. 1. Molecular Frameworks", *J. Med. Chem*, 39, 15, pp. 2887-2893.

Ben-Gal, I., A. Shani, A. Gohr, J. Grau, S. Arviv, A. Shmilovici, S. Posch, and I. Grosse

2005 "Identification of Transcription Factor Binding Sites with Variable-Order Bayesian Networks", *Bioinformatics*, 21, 11 (Mar. 2005), pp. 2657-2666.

Bender, Andreas, Jeremy L. Jenkins, Josef Scheiber, Sai Chetan K. Sukuru, Meir Glick, and John W. Davies

2009 "How Similar Are Similarity Searching Methods? A Principal Component Analysis of Molecular Descriptor Space", *J. Chem. Inf. Model.* 49, 1 (Jan. 2009), pp. 108-119.

Bengio, Y., P. Simard, and P. Frasconi

1994 "Learning Long-Term Dependencies with Gradient Descent is Difficult", *IEEE T. Neural Net.* 5, 2 (Mar. 1994), pp. 157-166.

Benjamini, Yoav and Yosef Hochberg

1995 "Controlling the False Discovery Rate: A Practical and Powerful Approach to Multiple Testing", *J. R. Stat. Soc. Series B Stat. Methodol.* Pp. 289-300.

Bento, Patricia A., Anna Gaulton, Anne Hersey, Louisa J. Bellis, Jon Chambers, Mark Davies, Felix A. Krüger, Yvonne Light, Lora Mak, Shaun McGlinchey, Michal Nowotka, George Papadatos, Rita Santos, and John P. Overington

2014 "The ChEMBL Bioactivity Database: An Update", *Nucleic Acids Res.* 42, D1 (Jan. 2014), pp. D1083-D1090.

Bitencourt-Ferreira, Gabriela and Walter Filgueira de Azevedo

2019 "Docking with SwissDock", in *Docking Screens for Drug Discovery*, Springer, pp. 189-202.

Bjerrum, Esben Jannik

2017 "SMILES Enumeration as Data Augmentation for Neural Network Modeling of Molecules", *arXiv*, 1703.07076 (Mar. 2017).

Bongers, Brandon J, Adriaan P Ijzerman, and Gerard JP Van Westen

2020 "Proteochemometrics Recent Developments in Bioactivity and Selectivity Modeling", *Drug Discov. Today: Technol.*

Born, Jannis, Matteo Manica, Ali Oskooei, Joris Cadow, and Karsten Borgwardt
2020 "PaccMann RL : Designing Anticancer Drugs from Transcriptomic Data via Reinforcement Learning", *arXiv*, 1909.05114v4.

Bredel, Markus and Edgar Jacoby
2004 "Chemogenomics: An Emerging Strategy for Rapid Target and Drug Discovery", *Nat. Rev. Genet.* 5, 4, pp. 262-275.

Breiman, Leo
2001 "Random Forests", *Mach. Learn.* 45, 1, pp. 5-32.

Brooks, Bernard R, Charles L Brooks, Alexander D MacKerell, Lennart Nilsson, Robert J Petrella, Benoit Roux, Youngdo Won, Georgios Archontis, Christian Bartels, Stefan Boresch, et al.
2009 "CHARMM: The Biomolecular Simulation Program", *J. Comput. Chem.* 30, 10, pp. 1545-1614.

Brown, Milton L., Scott Grindrod, Thomas H. Walls, Todd Hansen, Simeng Suy, and Mikell A. Paige
2014 "Na Channels, Disease, and Related Assays and Compositions", US 8816095.

Bruce, Craig L., James L. Melville, Stephen D. Pickett, and Jonathan D. Hirst
2007 "Contemporary QSAR Classifiers Compared", *J. Chem. Inf. Model.* 47, 1, pp. 219-227.

Buda, Mateusz, Atsuto Maki, and Maciej A Mazurowski
2018 "A Systematic Study of the Class Imbalance Problem in Convolutional Neural Networks", *Neural Networks*, 106, pp. 249-259.

Bull, Simon C. and Andrew J. Doig
2015 "Properties of Protein Drug Target Classes", *PLoS ONE*, 10, 3, pp. 1-44.

Cao, Dong Sheng, Yi Zeng Liang, Jun Yan, Gui Shan Tan, Qing Song Xu, and Shao Liu
2013 "PyDPI: Freely Available Python Package for Chemoinformatics, Bioinformatics, and Chemogenomics Studies", *J. Chem. Inf. Model.* 53, 11, pp. 3086-3096.

Chawla, Nitesh V, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer
2002 "SMOTE: Synthetic Minority Over-Sampling Technique", *J. Artif. Intell. Res.* 16, pp. 321-357.

Chen, Yu-Chian and J. Apostolakis
2015 "Beware of Docking!", *Trends Pharmacol. Sci.* 36, 2 (Feb. 2015), pp. 78-95.

Chen, Hongming, Ola Engkvist, Yinhai Wang, Marcus Olivecrona, and Thomas Blaschke
2018 "The Rise of Deep Learning in Drug Discovery", *Drug Discov. Today*, 23, 6 (June 2018), pp. 1241-1250.

Chen, Tianqi and Carlos Guestrin

2016 "XGBoost: A Scalable Tree Boosting System", in *Proc. 22nd ACM Int. Conf. Knowl. Discovery Data Min.* Pp. 785-794.

Chen, Tianqi, Tong He, Michael Benesty, Vadim Khotilovich, Yuan Tang, Hyunsu Cho, et al.

2015 "Xgboost: Extreme Gradient Boosting", *R package version 0.4-2*, 1, 4.

Chen, Wei, Hao Lin, and Kuo Chen Chou

2015 "Pseudo Nucleotide Composition or PseKNC: An Effective Formulation for Analyzing Genomic Sequences", *Mol. Biosyst.* 11, 10, pp. 2620-2634.

Chen, Xi, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel

2016 "InfoGAN: Interpretable Representation Learning by Information Maximizing Generative Adversarial Nets", in *Adv. Neur. In.* 1606.03657v1, pp. 2172-2180.

Chen, Zhen, Pei Zhao, Fuyi Li, Tatiana T Marquez-Lago, Andre Leier, Jerico Revote, Yan Zhu, David R Powell, Tatsuya Akutsu, Geoffrey I Webb, Kuo-Chen Chou, A Ian Smith, Roger J Daly, Jian Li, and Jiangning Song

2019 "iLearn: an Integrated Platform and Meta-Learner for Feature Engineering, Machine-Learning Analysis and Modeling of DNA, RNA and Protein Sequence Data", *Briefings Bioinf.* (Apr. 2019).

Cho, Kyunghyun, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio

2014 "Learning Phrase Representations Using RNN Encoder-Decoder for Statistical Machine Translation", *arXiv*, 1406.1078.

Chollet, François et al.

2015 *Keras*, `https://keras.io`, (accessed May 25, 2018).

Christen, Markus, Philippe H H Nenberger, Dirk Bakowies, Riccardo Baron, Roland B Rgi, Daan P Geerke, Tim N Heinz, Mika A Kastenholz, Vincent Kr Utler, Chris Oostenbrink, Christine Peter, Daniel Trzesniak, and Wilfred F Van Gunsteren

2005 "The GROMOS Software for Biomolecular Simulation: GROMOS05", *J. Comput. Chem.* 26, 16, pp. 1719-1751.

Chung, Junyoung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio

2014 "Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling", *arXiv*, 1412.3555, pp. 1-9.

Cohen, Jacob

1960 "A Coefficient of Agreement for Nominal Scales", *Educ. Psychol. Meas.* 20, 1, pp. 37-46.

Consortium, The UniProt

2018 "UniProt: a Worldwide Hub of Protein Knowledge", *Nucleic Acids Res.* 47, D1 (Nov. 2018), pp. D506-D515.

Cornell, Wendy D., Piotr Cieplak, Christopher I. Bayly, Ian R. Gould, Kenneth M. Merz, David M. Ferguson, David C. Spellmeyer, Thomas Fox, James W. Caldwell, and Peter A. Kollman

1996 "A Second Generation Force Field for the Simulation of Proteins, Nucleic Acids, and Organic Molecules", *J. Am. Chem. Soc.* 118, 9 (Jan. 1996), pp. 2309-2309.

Cortes, Corinna and Vladimir Vapnik

1995 "Support-Vector Networks", *Mach. Learn.* 20, pp. 273-297.

Crawford, Jake and Casey S. Greene

2020 "Incorporating Biological Structure into Machine Learning Models in Biomedicine", *Curr. Opin. Biotech.* 63, pp. 126-134.

Dahl, George E, Navdeep Jaitly, and Ruslan Salakhutdinov

2014 "Multi-task Neural Networks for QSAR Predictions", *arXiv*, 1406.1231.

Dakshanamurthy, Sivanesan, Naiem T. Issa, Shahin Assefnia, Ashwini Seshasayee, Oakland J. Peters, Subha Madhavan, Aykut Uren, Milton L. Brown, and Stephen W. Byers

2012 "Predicting New Indications for Approved Drugs Using a Proteochemometric Method", *J. Med. Chem.* 55, 15, pp. 6832-6848.

Daniel, Wayne W

1999 *Biostatistics: a Foundation for Analysis in the Health Sciences*, 9th ed., John Wiley and Sons, New York.

Deng, Jia, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei

2009 "ImageNet: A Large-Scale Hierarchical Image Database", in *2009 Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.* IEEE, pp. 248-255.

"Descriptors for Chemical Compounds"

2008 in *Handbook of Chemoinformatics*, Wiley-VCH Verlag GmbH, Weinheim, Germany, pp. 977-979, ISBN: 9783527618279.

Devlin, Jacob, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova

2018 "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding", *arXiv*, 1810.04805.

Di Lena, Pietro, Ken Nagata, and Pierre Baldi

2012 "Deep Architectures for Protein Contact Map Prediction", *Bioinformatics*, 28, 19, pp. 2449-2457.

Dieleman, Sander, Jan Schlüter, Colin Raffel, Eben Olson, Sören Kaae Sönderby, Daniel Nouri, Daniel Maturana, Martin Thoma, Eric Battenberg, Jack Kelly, Jeffrey De Fauw, Michael Heilman, Diogo Moitinho de Almeida, Brian McFee, Hendrik Weideman, Gábor Takács, Peter de Rivaz, Jon Crall, Gregory Sanders, Kashif Rasul, Cong Liu, Geoffrey French, and Jonas Degrave

2015 *Lasagne: First release.* http://dx.doi.org/10.5281/zenodo.27878.

DiMasi, Joseph A., Henry G. Grabowski, and Ronald W. Hansen

2016 "Innovation in the Pharmaceutical Industry: New Estimates of R & D Costs", *J. Health Econ.* 47, pp. 20-33.

Domingos, Pedro

2012 "A Few Useful Things to Know About Machine Learning", *Commun. ACM*, 55, 10, pp. 78-87.

Dong, Qi, Shaogang Gong, and Xiatian Zhu

2018 "Imbalanced Deep Learning by Minority Class Incremental Rectification", *IEEE Trans. Pattern Anal. Mach. Intell.* 41, 6, pp. 1367-1381.

Duchi, John, Elad Hazan, and Yoram Singer

2010 "Adaptive Subgradient Methods for Online Learning and Stochastic Optimization", *23rd COLT*, pp. 257-269.

Duvenaud, David K, Dougal Maclaurin, Jorge Iparraguirre, Rafael Bombarell, Timothy Hirzel, Alán Aspuru-Guzik, and Ryan P Adams

2015 "Convolutional Networks on Graphs for Learning Molecular Fingerprints", in *Adv. Neur. In.* Pp. 2224-2232.

Dwarampudi, Mahidhar and N V Subba Reddy

2019 "Effects of Padding on LSTMs and CNNs", *arXiv*, 1903.07288, 1903.07288.

Ekins, Sean

2016 "The Next Era: Deep Learning in Pharmaceutical Research", *Pharm. Res.* 33, 11 (Nov. 2016), pp. 2594-2603.

Elbasir, Abdurrahman, Balasubramanian Moovarkumudalvan, Khalid Kunji, Prasanna R. Kolatkar, Raghvendra Mall, and Halima Bensmail

2019 "Deepcrystal: A Deep Learning Framework for Sequence-Based Protein Crystallization Prediction", *Bioinformatics*, 35, 13, pp. 2216-2225.

Ellrott, Kyle, Christian M Zmasek, Dana Weekes, S Sri Krishna, Constantina Bakolitsa, Adam Godzik, and John Wooley

2011 "TOPSAN: a Dynamic Web Database for Structural Genomics", *Nucleic Acids Res.* 39, Database, pp. 494-496.

Eraslan, Gökcen, Žiga Avsec, Julien Gagneur, and Fabian J. Theis

2019 "Deep Learning: New Computational Modelling Techniques for Genomics", *Nat. Rev. Genet.* 20, 7, pp. 389-403.

Evans, R, J Jumper, J Kirkpatrick, L Sifre, T.F.G Green, C. Qin, A. Zidek, A Nelson, A. Bridgland, H. Penedones, S. Petersen, K. Simonyan, S Crossan, D.T. Jones, D Silver, K. Kavukcuoglu, D. Hassabis, and A.W. Senior

2018 "De Novo Structure Prediction with Deep Learning-Based Scoring", *13th CASP*.

Fellbaum, Christiane

1998 *WordNet: An Electronic Lexical Database*, Cambridge, MA: MIT Press.

Ferreira, Leonardo, Ricardo dos Santos, Glaucius Oliva, and Adriano Andricopulo

2015 "Molecular Docking and Structure-Based Drug Design Strategies", *Molecules*, 20, 7 (July 2015), pp. 13384-13421.

Fisher, Ronald Aylmer

1992 "Statistical Methods for Research Workers", in *Breakthroughs in statistics*, Springer, pp. 66-70.

Forli, Stefano, Ruth Huey, Michael E Pique, Michel Sanner, David S Goodsell, and Arthur J Olson

2016 "Computational Protein-Ligand Docking and Virtual Drug Screening with the AutoDock Suite", *Nat. Protoc.* 11, 5, pp. 905-919.

Fourches, Denis, Eugene Muratov, and Alexander Tropsha

2010 "Trust, but Verify: On the Importance of Chemical Structure Curation in Cheminformatics and QSAR Modeling Research", *J. Chem. Inf. Model.* 50, 7, pp. 1189-1204.

Friesner, Richard A, Jay L Banks, Robert B Murphy, Thomas A Halgren, Jasna J Klicic, Daniel T Mainz, Matthew P Repasky, Eric H Knoll, Mee Shelley, Jason K Perry, David E Shaw, Perry Francis, Peter S Shenkin, and D E Shaw Research

2004 "Glide: A New Approach for Rapid, Accurate Docking and Scoring. 1. Method and Assessment of Docking Accuracy", *J. Med. Chem.* 47, 7, pp. 1739-1749.

Frimurer, Thomas M., Trond Ulven, Christian E. Elling, Lars-Ole Gerlach, Evi Kostenis, and Thomas Högberg

2005 "A Physicogenetic Method to Assign Ligand-Binding Relationships Between 7TM Receptors", *Bioorg. Med. Chem. Lett.* 15, 16 (Aug. 2005), pp. 3707-3712.

Furnham, Nicholas, Ian Sillitoe, Gemma L. Holliday, Alison L. Cuff, Roman A. Laskowski, Christine A. Orengo, and Janet M. Thornton

2012 "Exploring the Evolution of Novel Enzyme Functions within Structurally Defined Protein Superfamilies", *PLoS Comput. Biol.* 8, 3 (Mar. 2012), e1002403.

Gang, Peng, Wang Zhen, Wei Zeng, Yuri Gordienko, Yuriy Kochura, Oleg Alienin, Oleksandr Rokovyi, and Sergii Stirenko

2018 "Dimensionality Reduction in Deep Learning for Chest X-ray Analysis of Lung Cancer", in *2018 10th ICACI*, IEEE, pp. 878-883.

García, Salvador, Sergio Ramírez-gallego, Julián Luengo, José Manuel Benítez, and Francisco Herrera

2016 "Big Data Preprocessing : Methods and Prospects", *Big Data Anal.* 1, 9.

Gawehn, Erik, Jan A. Hiss, and Gisbert Schneider

2016 "Deep Learning in Drug Discovery", *Mol. Inf.* 35, 1 (Jan. 2016), pp. 3-14.

Gers, Felix A, Jürgen Schmidhuber, and Fred Cummins

2000 "Learning to Forget : Continual Prediction with LSTM", *Neural Comput.* 12, 10, pp. 2451-2471.

Ghasemi, Fahimeh, Alireza Mehridehnavi, Afshin Fassihi, and Horacio Pérez-Sánchez

2017 "Deep Neural Network in Biological Activity Prediction using Deep Belief Network Deep neural network in QSAR studies using deep belief network", *Appl. Soft Comput. J.* 62, October, pp. 251-258.

Ghasemi, Fahimeh, Alireza Mehridehnavi, Alfonso Pérez-Garrido, and Horacio Pérez-Sánchez

2018 "Neural Network and Deep-Learning Algorithms Used in QSAR Studies: Merits and Drawbacks", *Drug Discov. Today*, 23, 10, pp. 1784-1790, ISSN: 18785832.

Gilson, Michael K., Tiqing Liu, Michael Baitaluk, George Nicola, Linda Hwang, Jenny Chong, Ohl P., Sieb C., Thiel K., Wiswedel B., Preisach C, Burkhardt H, Schmidt-Thieme L, and Decker R

2016 "BindingDB in 2015: A Public Database for Medicinal Chemistry, Computational Chemistry and Systems Pharmacology", *Nucleic Acids Res.* 44, D1 (Jan. 2016), pp. D1045-D1053.

Glorot, Xavier, Antoine Bordes, and Yoshua Bengio

2011 "Deep Sparse Rectifier Neural Networks", *Proc. Inter. Conf. Artif. Int. Stat.* Pp. 315-323.

Goh, Garrett B., Nathan O. Hodas, Charles Siegel, and Abhinav Vishnu

2017a "SMILES2Vec: An Interpretable General-Purpose Deep Neural Network for Predicting Chemical Properties", *arXiv*, 1712.02034.

Goh, Garrett B., Nathan O. Hodas, and Abhinav Vishnu

2017b "Deep learning for Computational Chemistry", *J. Comput. Chem.* 38, 16 (June 2017), pp. 1291-1307.

Goh, Garrett B., Charles Siegel, Abhinav Vishnu, Nathan O. Hodas, and Nathan Baker

2017c "Chemception: A Deep Neural Network with Minimal Chemistry Knowledge Matches the Performance of Expert-developed QSAR/QSPR Models", *arXiv*, 1706.06689 (June 2017).

Gomes, Joseph, Bharath Ramsundar, Evan N Feinberg, and Vijay S Pande

2017 "Atomic Convolutional Networks for Predicting Protein-Ligand Binding Affinity", *arXiv*, 1703.10603.

Goodfellow, Ian J, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio

2014 "Generative Adversarial Nets", in *Adv. Neur. Inf.* Pp. 2672-2680.

Granier, Sébastien and Brian Kobilka

2012 "A New Era of GPCR Structural and Chemical Biology", *Nat. Chem. Biol.* 8, 8, pp. 670-673.

Graves, Alex et al.
2012 *Supervised Sequence Labelling with Recurrent Neural Networks*, Springer, vol. 385.

Graves, Alex and Jürgen Schmidhuber
2005 "Framewise Phoneme Classification with Bidirectional LSTM and Other Neural Network Architectures", *Neural Networks*, 18, 5, pp. 602-610.

Grosdidier, Aurélien, Vincent Zoete, and Olivier Michielin
2011 "SwissDock, a Protein-Small Molecule Docking Web Service Based on EADock DSS." *Nucleic Acids Res.* 39, Web Server issue (July 2011), W270-7.

Guidotti, Riccardo, Anna Monreale, Salvatore Ruggieri, Franco Turini, Fosca Giannotti, and Dino Pedreschi
2018 "A Survey of Methods for Explaining Black Box Models", *ACM CSUR*, 51, 5, pp. 1-42.

Halder, Amit Kumar, Ana S. Moura, and M. Natalia D.S. Cordeiro
2018 "QSAR Modelling: a Therapeutic Patent Review 2010-Present", *Expert Opin. Ther. Pat.* 28, 6, pp. 467-476.

Hamanaka, Masatoshi, Kei Taneishi, Hiroaki Iwata, Jun Ye, Jianguo Pei, Jinlong Hou, and Yasushi Okuno
2017 "CGBVS-DNN: Prediction of Compound-protein Interactions Based on Deep Learning", *Mol. Inf.* 36, 1-2 (Jan. 2017), p. 1600045.

Hannun, Awni, Carl Case, Jared Casper, Bryan Catanzaro, Greg Diamos, Erich Elsen, Ryan Prenger, Sanjeev Satheesh, Shubho Sengupta, Adam Coates, and Andrew Y. Ng
2014 "Deep Speech: Scaling Up End-to-End Speech Recognition", *arXiv*, 1412.5567, pp. 1-12.

Hansch, Corwin and Toshio Fujita
1964 "p-σ-π Analysis. A Method for the Correlation of Biological Activity and Chemical Structure", *J. Am. Chem. Soc.* 86, 8, pp. 1616-1626.

Hardin, James W, James William Hardin, Joseph M Hilbe, and Joseph Hilbe
2007 *Generalized Linear Models and Extensions*, Stata Press.

He, Kaiming, Xiangyu Zhang, Shaoqing Ren, and Jian Sun
2016 "Deep Residual Learning for Image Recognition", in *2016 Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.* IEEE, pp. 770-778.

Heikamp, Kathrin and Jürgen Bajorath
2011 "Large-Scale Similarity Search Profiling of ChEMBL Compound Data Sets", *J. Chem. Inf. Model.* 51, 8 (Aug. 2011), pp. 1831-1839.

Hert, Jerome, Peter Willett, David J Wilton, Pierre Acklin, Kamal Azzaoui, Edgar Jacoby, and Ansgar Schuffenhauer
2004 "Comparison of Topological Descriptors for Similarity-Based Virtual Screening Using Multiple Bioactive Reference Structures." *Org. Biomol. Chem.* 2, 22 (Nov. 2004), pp. 3256-66, ISSN: 1477-0520.

Hinton, Geoffrey E, Simon Osindero, and Yee-Whye Teh

2006 "A Fast Learning Algorithm for Deep Belief Nets", *Neural Comput.* 18, 7, pp. 1527-1554.

Hinton, Geoffrey, Nitish Srivastava, and Kevin Swersky

2012 *Neural Networks for Machine Learning Lecture 6a Overview of Mini-Batch Gradient Descent*.

Hlavac, Marek

2018 *stargazer: Well-Formatted Regression and Summary Statistics Tables*, R package version 5.2.2, Central European Labour Studies Institute (CELSI), Bratislava, Slovakia, https://CRAN.R-project.org/package=stargazer.

Hochreiter, Sepp, Yoshua Bengio, Paolo Frasconi, and Jürgen Schmidhuber

2001 "Gradient Flow in Recurrent Nets: the Difficulty of Learning Long-Term Dependencies", in *A field guide to dynamical recurrent neural networks. IEEE Press*.

Hochreiter, Sepp, Martin Heusel, and Klaus Obermayer

2007 "Fast Model-Based Protein Homology Detection Without Alignment", *Bioinformatics*, 23, 14 (July 2007), pp. 1728-1736.

Hochreiter, Sepp and Jürgen Schmidhuber

1997 "Long Short-Term Memory", *Neural Comput.* 9, pp. 1735-1780.

Hollander, Myles and Douglas A Wolfe

1999 *Nonparametric Statistical Methods*, 2nd ed., John Wiley & Sons, New York.

Holliday, Gemma L., Claudia Andreini, Julia D. Fischer, Syed Asad Rahman, Daniel E. Almonacid, Sophie T. Williams, and William R. Pearson

2012 "MACiE: Exploring the Diversity of Biochemical Reactions", *Nucleic Acids Res.* 40, D1, pp. 783-789.

Honda, Shion, Shoi Shi, and Hiroki R. Ueda

2019 "SMILES Transformer: Pre-trained Molecular Fingerprint for Low Data Drug Discovery", *arXiv*, 1911.04738.

Hopfield, John J

1982 "Neural Networks and Physical Systems with Emergent Collective Computational Abilities", *Proc. Natl. Acad. Sci.* 79, 8, pp. 2554-2558.

Hoppe, Christian, Christoph Steinbeck, and Gerd Wohlfahrt

2006 "Classification and Comparison of Ligand-Binding Sites Derived from Grid-Mapped Knowledge-Based Potentials", *J. Mol. Graph. Model.* 24, pp. 328-340.

Hs Segler, Marwin, Thierry Kogej, Christian Tyrchan, and Mark P Waller

2017 "Generating Focussed Molecule Libraries for Drug Discovery with Recurrent Neural Networks", *arXiv*, 1701.01329v1.

Hsieh, Jui-Hua, Xiang S. Wang, Denise Teotico, Alexander Golbraikh, and Alexander Tropsha

2008 "Differentiation of AmpC Beta-Lactamase Binders vs. Decoys Using Classification kNN QSAR Modeling and Application of the QSAR Classifier to Virtual Screening", *J. Comput. Aid. Mol. Des.* 22, 9 (Sept. 2008), pp. 593-609.

Huang, Chen, Yining Li, Chen Change Loy, and Xiaoou Tang

2016 "Learning Deep Representation for Imbalanced Classification", in *Proc. IEEE Conf. CVPR*, pp. 5375-5384.

Huang, Niu, Brian K. Shoichet, and John J. Irwin

2006 "Benchmarking Sets for Molecular Docking", *J. Med. Chem.* 49, 23 (Nov. 2006), pp. 6789-6801.

Huang, Qi, Haixiao Jin, Qi Liu, Qiong Wu, Hong Kang, Zhiwei Cao, and Ruixin Zhu

2012 "Proteochemometric Modeling of the Bioactivity Spectra of HIV-1 Protease Inhibitors by Introducing Protein-Ligand Interaction Fingerprint", *PLoS ONE*, 7, 7, pp. 1-8.

Hughes, Tyler B., Grover P. Miller, and S. Joshua Swamidass

2015a "Modeling Epoxidation of Drug-like Molecules with a Deep Machine Learning Network", *ACS Cent. Sci.* 1, 4 (July 2015), pp. 168-180.

2015b "Site of Reactivity Models Predict Molecular Reactivity of Diverse Chemicals with Glutathione", *Chem. Res. Toxicol.* 28, 4 (Apr. 2015), pp. 797-809.

Id, Ingoo Lee, Jongsoo Keum, and Hojung Nam Id

2019 "DeepConv-DTI : Prediction of Drug-Target Interactions via Deep Learning with Convolution on Protein Sequences", *PLoS Comput. Biol.* 15, 6, e1007129.

Imming, Peter, Christian Sinning, and Achim Meyer

2006 "Drugs, their Targets and the Nature and Number of Drug Targets." *Nat. Rev. Drug Discov.* 5, 10, pp. 821-34.

Ing, HR

1964 "The Pharmacology of Homologous Series", in *Fortschritte der Arzneimittelforschung/Progress in Drug Research*, Springer, pp. 305-339.

Ioffe, Sergey and Christian Szegedy

2015 "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift", in pp. 448-456.

Irwin, John J., Brian K. Shoichet, Michael M. Mysinger, Niu Huang, Francesco Colizzi, Pascal Wassam, and Yiqun Cao

2009 "Automated Docking Screens: A Feasibility Study", *J. Med. Chem.* 52, 18 (Sept. 2009), pp. 5712-5720.

Irwin, John J., Teague Sterling, Michael M. Mysinger, Erin S. Bolstad, and Ryan G. Coleman

2012 "ZINC: A Free Tool to Discover Chemistry for Biology", *J. Chem. Inf. Model.* 52, 7 (July 2012), pp. 1757-1768.

Isola, Phillip, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros

2016 "Image-to-Image Translation with Conditional Adversarial Networks", *arXiv*, 1611.07004v1.

Jaeger, Sabrina, Simone Fulle, and Samo Turk

2018 "Mol2vec: Unsupervised Machine Learning Approach with Chemical Intuition", *J. Chem. Inf. Model.* 58, 1, pp. 27-35.

Jastrzebski, Stanislaw, Damian Lesniak, and Wojciech Marian Czarnecki

2018 "Learning to SMILE(S)", *arXiv*, 1602.06289v2.

Jeske, Lisa, Sandra Placzek, Ida Schomburg, Antje Chang, and Dietmar Schomburg

2018 "BRENDA in 2019: a European ELIXIR core data resource", *Nucleic Acids Res.* 47, D1 (Nov. 2018), pp. D542-D549.

Jimenez, Jose, Miha Skalic, Gerard Martinez-Rosell, and Gianni De Fabritiis

2018 "KDEEP: Protein–ligand Absolute Binding Affinity Prediction via 1D-Convolutional Neural Networks", *J. Chem. Inf. Model.* 58, 2, pp. 287-296.

Jing, Xiaoyang, Qiwen Dong, DC HONG, and Ruqian Lu

2019 "Amino Acid Encoding Methods for Protein Sequences: a Comprehensive Review and Assessment", *IEEE-ACM T. Comput. Bi.* PP, c, pp. 1-1.

Johnson, Justin M and Taghi M Khoshgoftaar

2019 "Survey on Deep Learning with Class Imbalance", *J. Big Data*, 6, 27.

Jones, William, Kaur Alasoo, Dmytro Fishman, and Leopold Parts

2017 "Computational Biology: Deep Learning", *Emerging Top. Life Sci.* 1, 3, pp. 257-274.

Jurtz, Vanessa Isabell, Alexander Rosenberg Johansen, Morten Nielsen, Jose Juan Almagro Armenteros, Henrik Nielsen, Casper Kaae Sönderby, Ole Winther, and Sören Kaae Sönderby

2017 "An Introduction to Deep Learning on Biological Sequence Data: Examples and Solutions", *Bioinformatics*, 33, 22 (Nov. 2017), pp. 3685-3690.

Kaae Sönderby, Sören and Ole Winther

2015 "Protein Secondary Structure Prediction with Long Short Term Memory Networks", *arXiv*, 1412.7828v2.

Kadurin, Artur, Alexander Aliper, Andrey Kazennov, Polina Mamoshina, Quentin Vanhaelen, Kuzma Khrabrov, and Alex Zhavoronkov

2017a "The Cornucopia of Meaningful Leads: Applying Deep Adversarial Autoencoders for New Molecule Development in Oncology", *Oncotarget*, 8, 7, pp. 10883-10890.

Kadurin, Artur, Sergey Nikolenko, Kuzma Khrabrov, Alex Aliper, and Alex Zhavoronkov

2017b "druGAN: An Advanced Generative Adversarial Autoencoder Model for de Novo Generation of New Molecules with Desired Molecular Properties in Silico", *Mol. Pharm.* 14, 9 (Sept. 2017), pp. 3098-3104.

Karimi, Mostafa, Di Wu, Zhangyang Wang, and Yang Shen

2019 "DeepAffinity: Interpretable Deep Learning of Compound-Protein Affinity through Unified Recurrent and Convolutional Neural Networks", *Bioinformatics*, 35, 18, pp. 3329-3338.

Kauderer-Abrams, Eric

2017 "Quantifying Translation-Invariance in Convolutional Neural Networks", *arXiv*, 1801.01450.

Kearnes, Steven, Brian Goldman, and Vijay Pande

2016 "Modeling Industrial ADMET Data with Multitask Networks", *arXiv*, 1606.08793 (June 2016).

2017 "Modeling Industrial ADMET Data with Multitask Networks", *arXiv*, 1606.08793.

Kelley, David R, Jasper Snoek, and John L Rinn

2016 "Basset: Learning the Regulatory Code of the Accessible Genome with Deep Convolutional Neural Networks", *Genome Res.* 26, 7, pp. 990-999.

Khan, Salman H, Munawar Hayat, Mohammed Bennamoun, Ferdous A Sohel, and Roberto Togneri

2017 "Cost-Sensitive Learning of Deep Feature Representations from Imbalanced Data", *IEEE Trans. Neural Netw. Learn. Syst.* 29, 8, pp. 3573-3587.

Kim, Sunghwan, Jie Chen, Tiejun Cheng, Asta Gindulyte, Jia He, Siqian He, Qingliang Li, Benjamin A Shoemaker, Paul A Thiessen, Bo Yu, Leonid Zaslavsky, Jian Zhang, and Evan E Bolton

2019 "PubChem 2019 Update: Improved Access to Chemical Data", *Nucleic Acids Res.* 47, D1 (Oct. 2019), pp. D1102-D1109.

Kimber, Talia B, Sebastian Engelke, Igor V Tetko, Eric Bruno, and Guillaume Godin

2018 "Synergy Effect Between Convolutional Neural Networks and the Multiplicity of SMILES for Improvement of Molecular Prediction", *arXiv preprint arXiv:1812.04439*.

Kimothi, Dhananjay, Akshay Soni, Pravesh Biyani, and James M Hogan

2016 "Distributed Representations for Biological Sequence Analysis", *arXiv*, 1608.05949v2.

Kingma, Diederik P. and Max Welling

2014 "Auto-Encoding Variational Bayes", *Conf. Proc. 2nd ICLR*, Ml, pp. 1-14.

Kingma, Diederik P and Jimmy Lei Ba

2014 "Adam: A Method For Stochastic Optimization", *arXiv*, 1412.6980.

Klabunde, Thomas

2007 "Chemogenomic Approaches to Drug Discovery: Similar Receptors Bind Similar Ligands." *Brit. J. Pharmacol.* 152, 1 (Sept. 2007), pp. 5-7.

Kontijevskis, Aleksejs, Jan Komorowski, and Jarl E. S. Wikberg

2008 "Generalized Proteochemometric Model of Multiple Cytochrome P450 Enzymes and Their Inhibitors", *J. Chem. Inf. Model.* 48, 9 (Sept. 2008), pp. 1840-1850.

Korkmaz, Selcuk

2020 "Deep Learning-Based Imbalanced Data Classification for Drug Discovery", *J. Chem. Inf. Model.* ISSN: 1549-9596.

Korotcov, Alexandru, Valery Tkachenko, Daniel P. Russo, and Sean Ekins

2017 "Comparison of Deep Learning with Multiple Machine Learning Methods and Metrics Using Diverse Drug Discovery Data Sets", *Mol. Pharm.* 14, 12, pp. 4462-4475.

Köse, Timur, Su Özgür, Erdal Coşgun, Ahmet Keskinoğlu, and Pembe Keskinoğlu

2020 "Effect of Missing Data Imputation on Deep Learning Prediction Performance for Vesicoureteral Reflux and Recurrent Urinary Tract Infection Clinical Study", *BioMed Res. Int.* 2020.

Koutsoukas, Alexios, Keith J. Monaghan, Xiaoli Li, and Jun Huan

2017 "Deep-Learning: Investigating Deep Neural Networks Hyper-Parameters and Comparison of Performance to Shallow Methods for Modeling Bioactivity Data", *J. Cheminf.* 9, 1 (Dec. 2017), p. 42.

Kramer, Christian and Peter Gedeck

2011 "Global Free Energy Scoring Functions Based on Distance-Dependent Atom-Type Pair Descriptors", *J. Chem. Inf. Model.* 51, 3, pp. 707-720.

Krause, Karl Heinz, David Lambeth, and Martin Krönke

2013 *Why Enzymes as Drug Targets?*, Second Edi, John Wiley & Sons, Inc., pp. 1-23.

Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E Hinton

2012 "ImageNet Classification with Deep Convolutional Neural Networks", in Curran Associates, Inc., pp. 1097-1105.

Kuhn, Michael, Christian von Mering, Monica Campillos, Lars Juhl Jensen, and Peer Bork

2007 "STITCH: Interaction Networks of Chemicals and Proteins", *Nucleic Acids Res.* 36, suppl_1, pp. D684-D688.

Kumari, Indu, Padmani Sandhu, Mushtaq Ahmed, and Yusuf Akhter

2017 "Molecular Dynamics Simulations, Challenges and Opportunities: A Biologist's Prospective", *Curr. Protein Pept. Sc.* 18, 11.

Kutlushina, Alina, Aigul Khakimova, Timur Madzhidov, and Pavel Polishchuk

2018 "Ligand-Based Pharmacophore modeling Using Novel 3D Pharmacophore Signatures", *Molecules*, 23, 12, pp. 1-14.

Kwon, Sunyoung and Sungroh Yoon

2017 "DeepCCI: End-to-end Deep Learning for Chemical-Chemical Interaction Prediction", *Proc. Int. Conf. on Bioinformat. Comput. Biol. and Health Inform.* Pp. 203-212.

Lanchantin, Jack, Ritambhara Singh, Beilun Wang, and Yanjun Qi

2016 "Deep Motif Dashboard: Visualizing and Understanding Genomic Sequences Using Deep Neural Networks", *arXiv*, 1608.03644 (Aug. 2016).

Lapins, Maris, Martin Eklund, Ola Spjuth, Peteris Prusis, and Jarl E S Wikberg

2008 "Proteochemometric Modeling of HIV Protease Susceptibility", *BMC Bioinformatics*, 9, 1, p. 181.

Lapins, Maris and Jarl E. S. Wikberg

2009 "Proteochemometric Modeling of Drug Resistance over the Mutational Space for Multiple HIV Protease Variants and Multiple Protease Inhibitors", *J. Chem. Inf. Model.* 49, 5 (May 2009), pp. 1202-1210.

Lapinsh, Maris, Peteris Prusis, Alexandrs Gutcaits, Torbjörn Lundstedt, and Jarl ES Wikberg

2001 "Development of Proteochemometrics: a Novel Technology for the Analysis of Drug-Receptor Interactions", *BBA - Gen. Subjects*, 1525, 1, pp. 180-190.

Lapinsh, Maris, Peteris Prusis, Ilze Mutule, Felikss Mutulis, and Jarl E.S. Wikberg

2003 "QSAR and Proteochemometric Analysis of the Interaction of a Series of Organic Compounds with Melanocortin Receptor Subtypes", *J. Med. Chem.* 46, 13, pp. 2572-2579.

Law, Vivian, Craig Knox, Yannick Djoumbou, Tim Jewison, An Chi Guo, Yifeng Liu, Adam Maciejewski, David Arndt, Michael Wilson, Vanessa Neveu, Alexandra Tang, Geraldine Gabriel, Carol Ly, Sakina Adamjee, Zerihun T. Dame, Beomsoo Han, You Zhou, and David S. Wishart

2014 "DrugBank 4.0: Shedding New Light on Drug Metabolism", *Nucleic Acids Res.* 42, D1 (Jan. 2014), pp. D1091-D1097.

LeCun, Yann, Yoshua Bengio, and Geoffrey Hinton

2015 "Deep Learning", *Nature*, 521, 7553, pp. 436-444.

LeCun, Yann, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel

1989 "Backpropagation Applied to Handwritten Zip Code Recognition", *Neural Comput.* 1, 4, pp. 541-551.

LeCun, Yann, Léon Bottou, Yoshua Bengio, and Patrick Haffner
1998 "Gradient-Based Learning Applied to Document Recognition", *Proc. IEEE*, 86, 11, pp. 2278-2324.

Ledig, Christian, Lucas Theis, Ferenc Huszár, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, and Wenzhe Shi
2016 "Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network", *arXiv*, 1609.04802v3.

Lee, Hansang, Minseok Park, and Junmo Kim
2016 "Plankton Classification on Imbalanced Large Scale Database via Convolutional Neural Networks with Transfer Learning", in *2016 IEEE ICIP*, IEEE, pp. 3713-3717.

Lee, Jinhyuk, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, and Jaewoo Kang
2019 "BioBERT: a Pre-trained Biomedical Language Representation Model for Biomedical Text Mining", *Bioinformatics*, September, pp. 1-7.

Lei, Na, Kehua Su, Li Cui, Shing-Tung Yau, and Xianfeng David Gu
2019 "A Geometric View of Optimal Transportation and Generative Model", *Comput. Aided Geom. D.* 68, pp. 1-21.

Lemaître, Guillaume, Fernando Nogueira, and Christos K Aridas
2017 "Imbalanced-learn: A Python Toolbox to Tackle the Curse of Imbalanced Datasets in Machine Learning", *J. Mach. Learn. Res.* 18, 1, pp. 559-563.

Lenselink, Eelke B, Niels Ten Dijke, Brandon Bongers, George Papadatos, Herman W T van Vlijmen, Wojtek Kowalczyk, Adriaan P IJzerman, and Gerard J P van Westen
2017 "Beyond the Hype: Deep Neural Networks Outperform Established Methods Using a ChEMBL Bioactivity Benchmark Set", *J. Cheminf.* 9, 1 (Aug. 2017), p. 45.

Li, Hongjian, Kwong-Sak Leung, Pedro J. Ballester, Man-Hon Wong, and R Wang
2014 "istar: A Web Platform for Large-Scale Protein-Ligand Docking", *PLoS ONE*, 9, 1 (Jan. 2014), e85678.

Li, Ze-Rong, Hong Huang Lin, LY Han, L Jiang, X Chen, and Yu Zong Chen
2006 "PROFEAT: a Web Server for Computing Structural and Physicochemical Features of Proteins and Peptides from Amino Acid Sequence", *Nucleic Acid Res.* 34, suppl_2, W32-W37.

Li, Shuai, Wanqing Li, Chris Cook, Ce Zhu, and Yanbo Gao
2018 "Independently Recurrent Neural Network (IndRNN): Building a Longer and Deeper RNN", in *Proc. CVPR IEEE*, pp. 5457-5466.

Li, Yi, Daniel Quang, and Xiaohui Xie
2017 "Understanding Sequence Conservation With Deep Learning", *Proc. 8th ACM-BCB*, pp. 400-406.

Li, Yu, Sheng Wang, Ramzan Umarov, Bingqing Xie, Ming Fan, Lihua Li, Xin Gao, and John Hancock

2018 "DEEPre: Sequence-Based Enzyme EC Number Prediction by Deep Learning", *Bioinformatics*, 34, 5 (Mar. 2018), pp. 760-769.

Lipinski, Christopher A

2000 "Drug-like Properties and the Causes of Poor Solubility and Poor Permeability", *J. Pharmacol. Tox. Met.* 44, 1, pp. 235-249.

Lipton, Zachary C, John Berkowitz, and Charles Elkan

2015 "A Critical Review of Recurrent Neural Networks for Sequence Learning", *arXiv*, 1506.00019.

Liu, Bin, Xin Gao, and Hanyu Zhang

2019 "BioSeq-Analysis2.0: an Updated Platform for Analyzing DNA, RNA and Protein Sequences at Sequence Level and Residue Level Based on Machine Learning Approaches", *Nucleic Acids Res.* 47, 20 (Sept. 2019), e127-e127.

Liu, Bin, Fule Liu, Longyun Fang, Xiaolong Wang, and Kuo-Chen Chou

2014 "repDNA: a Python Package to Generate Various Modes of Feature Vectors for DNA Sequences by Incorporating User-efined Physico-chemical Properties and Sequence-order Effects", *Bioinformatics*, 31, 8 (Dec. 2014), pp. 1307-1309.

Lo, Yu-chen, Stefano E Rensi, Wen Torng, and Russ B Altman

2018 "Machine Learning in Chemoinformatics and Drug Discovery", *Drug Discov. Today*, 23, 8, pp. 1538-1546.

Lombardo, Franco, R Scott Obach, Marina Y Shalaeva, and Feng Gao

2004 "Prediction of Human Volume of Distribution Values for Neutral and Basic Drugs. 2. Extended Data Set and Leave-Class-Out Statistics", *J. Med. Chem.* 47, 5, pp. 1242-1250.

Lopez-Del Rio, Angela, Maria Martin, Alexandre Perera-Lluna, and Rabie Saidi

2020 "Effect of Sequence Padding on the Performance of Protein-Based Deep Learning Models", *Sci. Rep.* 10, p. 14634, DOI: 10.1038/s41598-020-71450-8.

Lopez-Del Rio, Angela, Alfons Nonell-Canals, David Vidal, and Alexandre Perera-Lluna

2019 "Evaluation of Cross-Validation Strategies in Sequence-Based Binding Prediction Using Deep Learning", *J. Chem. Inf. Model.* 59, 4, pp. 1645-1657, DOI: 10.1021/acs.jcim.8b00663.

Lopez-del Rio, Angela, Sergio Picart-Armada, and Alexandre Perera-Lluna

2021 "Balancing Data on Deep Learning-Based Proteochemometric Activity Classification", *J. Chem. Inf. Model.* 61, 4, pp. 1657-1669, DOI: 10.1021/acs.jcim.1c00086.

Lovino, Marta, Gianvito Urgese, Enrico Macii, Santa Di Cataldo, and Elisa Ficarra

2019 "A Deep Learning Approach to the Screening of Oncogenic Gene Fusions in Humans", *Int. J. Mol. Sci.* 20, 7, pp. 1-13.

Luong, Minh Thang, Hieu Pham, and Christopher D. Manning

2015 "Effective Approaches to Attention-Based Neural Machine Translation", *Conf. Proc. EMNLP*, 1508.04025, pp. 1412-1421.

Lusci, Alessandro, Gianluca Pollastri, and Pierre Baldi

2013 "Deep Architectures and Deep Learning in Chemoinformatics: the Prediction of Aqueous Solubility for Drug-Like Molecules." *J. Chem. Inf. Model.* 53, 7 (July 2013), pp. 1563-75.

Ma, Junshui, Robert P. Sheridan, Andy Liaw, George E. Dahl, and Vladimir Svetnik

2015 "Deep Neural Nets as a Method for Quantitative Structure–Activity Relationships", *J. Chem. Inf. Model.* 55, 2 (Feb. 2015), pp. 263-274.

Manica, Matteo, Ali Oskooei, Jannis Born, Vigneshwari Subramanian, Julio Saéz-Rodríguez, and Mariá Rodríguez Martínez

2019 "Toward Explainable Anticancer Compound Sensitivity Prediction via Multimodal Attention-Based Convolutional Encoders", *Mol. Pharm.* 2016.

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng

2015 *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*, http://tensorflow.org/.

Masko, David and Paulina Hensman

2015 *The Impact of Imbalanced Training Data for Convolutional Neural Networks*.

Mauri, Andrea, Viviana Consonni, Manuela Pavan, and Roberto Todeschini

2006 "Dragon Software: An Easy Approach to Molecular Descriptor Calculations", *Match*, 56, 2, pp. 237-248.

Mayr, Andreas, Günter Klambauer, Thomas Unterthiner, and Sepp Hochreiter

2016 "DeepTox: Toxicity Prediction using Deep Learning", *Front. Environ. Sci.* 3, p. 80.

Mayr, Andreas, Günter Klambauer, Thomas Unterthiner, Marvin Steijaert, Jörg K Wegner, Hugo Ceulemans, Djork-Arné Clevert, and Sepp Hochreiter

2018 "Large-Scale Comparison of Machine Learning Methods for Drug Target Prediction on ChEMBL", *Chem. Sci.* 9, p. 5441.

Mazzaferro, Carlo

2017 "Predicting Protein Binding Affinity With Word Embeddings and Recurrent Neural Networks", *bioRxiv*.

McCulloch, Warren and Walter Pitts

1943 "A Logical Calculus of the Ideas Immanent in Nervous Activity", *Bull. Math. Biophys.* 5, 4, pp. 115-133.

Medina-Franco, Jose Luis, Alexander Golbraikh, Scott Oloff, Rafael Castillo, and Alexander Tropsha

2005 "Quantitative Structure–Activity Relationship Analysis of Pyridinone HIV-1 Reverse Transcriptase Inhibitors using the k Nearest Neighbor Method and QSAR-based Database Mining", *J. Comput. Aid. Mol. Des.* 19, 4 (Apr. 2005), pp. 229-242.

Mehta, Parmita, Stephen Portillo, Magdalena Balazinska, and Andrew Connolly

2020 "Sampling for Deep Learning Model Diagnosis", *arXiv*, 2002.09754v1.

Mendez, David, Anna Gaulton, A. Patricia Bento, Jon Chambers, Marleen De Veij, Eloy Félix, María Paula Magariños, Juan F. Mosquera, Prudence Mutowo, Michal Nowotka, María Gordillo-Marañón, Fiona Hunter, Laura Junco, Grace Mugumbate, Milagros Rodriguez-Lopez, Francis Atkinson, Nicolas Bosc, Chris J. Radoux, Aldo Segura-Cabrera, Anne Hersey, and Andrew R. Leach

2019 "ChEMBL: Towards Direct Deposition of Bioassay Data", *Nucleic Acids Res.* 47, D1, pp. D930-D940.

Merck Activity Competition

2012 (accessed May 25, 2018), https://www.kaggle.com/c/MerckActivity.

Meysman, Pieter, Kathleen Marchal, and Kristof Engelen

2012 "DNA Structural Properties in the Classification of Genomic Transcription Regulation Elements", *Bioinf. Biol. Insights*, 6, BBI-S9426.

Miao, Yajie, Mohammad Gowayyed, and Florian Metze

2015 "EESEN: End-to-End Speech Recognition Using Deep RNN Models and WFST-Based Decoding", in *IEEE Workshop ASRU*, IEEE, pp. 167-174.

Min, Seonwoo, Byunghan Lee, and Sungroh Yoon

2017 "Deep Learning in Bioinformatics", *Brief. Bioinform.* 18, 5, pp. 851-869.

Mirabello, Claudio and Björn Wallner

2019 "RAWMSA: End-to-End Deep Learning Using Raw Multiple Sequence Alignments", *PLoS ONE*, 14, 8.

Mlinarić, Ana, Martina Horvat, and Vesna Šupak Smolčić

2017 "Dealing with the positive publication bias: Why you should really publish your negative results", *Biochemia medica: Biochemia medica*, 27, 3, pp. 447-452.

Morris, Garrett M, David S Goodsell, Robert S Halliday, Ruth Huey, William E Hart, Richard K Belew, and Arthur J Olson

1998 "Automated Docking Using a Lamarckian Genetic Algorithm and an Empirical Binding Free Energy Function", *J. Comput. Chem.* 19, 28, pp. 1639-1662.

Morris, Garrett M, Ruth Huey, William Lindstrom, Michel F Sanner, Richard K Belew, David S Goodsell, and Arthur J Olson

2009 "AutoDock4 and AutoDockTools4: Automated Docking with Selective Receptor Flexibility." *J. Comput. Chem.* 30, 16 (Dec. 2009), pp. 2785-91.

Müller, Alex T., Jan A. Hiss, and Gisbert Schneider

2018 "Recurrent Neural Network Model for Constructive Peptide Design", *J. Chem. Inf. Model.* 2, 58, pp. 472-479.

Mysinger, Michael M., Michael Carchia, John. J. Irwin, and Brian K. Shoichet

2012 "Directory of Useful Decoys, Enhanced (DUD-E): Better Ligands and Decoys for Better Benchmarking", *J. Med. Chem.* 55, 14 (July 2012), pp. 6582-6594.

Naumann, Thorsten and Hans Matter

2002 "Structural Classification of Protein Kinases Using 3D Molecular Interaction Field Analysis of their Ligand Binding Sites: Target Family Landscapes", *J. Med. Chem.* 45, 12, pp. 2366-2378.

Neil, Daniel, Marwin Segler, Laura Guasch, Mohamed Ahmed, Dean Plumbley, Matthew Sellwood, and Nathan Brown

2018 "Exploring Deep Recurrent Models with Reinforcement Learning for Molecule Design", *Workshop track - ICLR 2018*, pp. 1-15.

Ng, Patrick

2017 "dna2vec: Consistent Vector Representations of Variable-Length k-Mers", *arXiv*, 1701.06279 (Jan. 2017).

Nicholson, Jeremy K., John Connelly, John C. Lindon, and Elaine Holmes

2002 "Metabonomics: a Platform for Studying Drug Toxicity and Gene function", *Nat. Rev. Drug Discov.* 1, 2, pp. 153-161.

Nilakantan, Ramaswamy, Norman Bauman, J. Scott Dixon, and R. Venkataraghavan

1987 "Topological Torsion: A New Molecular Descriptor for SAR Applications. Comparison with Other Descriptors", *J. Chem. Inf. Model*, 27, 2 (May 1987), pp. 82-85.

O'Boyle, Noel M, Michael Banck, Craig A James, Chris Morley, Tim Vandermeersch, and Geoffrey R Hutchison

2011 "Open Babel: an Open Chemical Toolbox", *J. Cheminf.* 3, 1, p. 33.

O'Boyle, Noel M, Chris Morley, and Geoffrey R Hutchison
2008 "Pybel: a Python Wrapper for the OpenBabel Cheminformatics Toolkit", *Chem. Cent. J.* 2, 1, p. 5.

O'Donovan, Claire, Rolf Apweiler, and Amos Bairoch
2001 "The Human Proteomics Initative (HPI)", *Trends Biotechnol.* 19, 5, pp. 178-181.

Oloff, Scott, Richard B Mailman, and Alexander Tropsha
2005 "Application of Validated QSAR Models of D1 Dopaminergic Antagonists for Database Mining", *J. Med. Chem.* 48, 23, pp. 7322-7332.

Öztürk, Hakime, Arzucan Özgür, and Elif Ozkirimli
2018a "DeepDTA: Deep Drug-Target Binding Affinity Prediction", *Bioinformatics*, 34, 17, pp. i821-i829.

Öztürk, Hakime, Elif Ozkirimli, and Arzucan Özgür
2018b "A Novel Methodology on Distributed Representations of Proteins Using their Interacting Ligands", *Bioinformatics*, 34, 13, pp. i295-i303.

Pan, Xiaoyong, Peter Rijnbeek, Junchi Yan, and Hong-Bin Shen
2018 "Prediction of RNA-Protein Sequence and Structure Binding Preferences Using Deep Convolutional and Recurrent Neural Networks", *BMC Genom.* 19, 1, p. 511.

Pan, Xiaoyong and Hong-Bin Shen
2018 "Predicting RNA–protein Binding Sites and Motifs through Combining Local and Global Deep Convolutional Neural Networks", *Bioinformatics*, 34, 20 (May 2018), pp. 3427-3436.

Paricharak, Shardul, Isidro Cortes-Ciriano, Adriaan P. Ijzerman, Thérèse E. Malliavin, and Andreas Bender
2015 "Proteochemometric Modelling Coupled to In Silico Target Prediction: An Integrated Approach for the Simultaneous Prediction of Polypharmacology and Binding Affinity/Potency of Small Molecules", *J. Cheminf.* 7, 1, pp. 1-11.

Parikh, Ankur P, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit
2016 "A Decomposable Attention Model for Natural Language Inference", *arXiv*, 1606.01933.

Park, Yongjin and Manolis Kellis
2015 "Deep Learning for Regulatory Genomics", *Nat. Biotechnol.* 33, 1, pp. 825-826.

Paszke, Adam, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala

2019 "PyTorch: An Imperative Style, High-Performance Deep Learning Library", in *Advances in Neural Information Processing Systems 32*, Curran Associates, Inc., pp. 8024-8035.

Pérot, Stéphanie, Olivier Sperandio, Maria A. Miteva, Anne Claude Camproux, and Bruno O. Villoutreix

2010 "Druggable Pockets and Binding Site Centric Chemical Space: A Paradigm Shift in Drug Discovery", *Drug Discov. Today*, 15, 15-16, pp. 656-667.

Picart-Armada, Sergio, Steven J Barrett, David R Willé, Alexandre Perera-Lluna, Alex Gutteridge, and Benoit H Dessailly

2019 "Benchmarking Network Propagation Methods for Disease Gene Identification", *PLoS Comput. Biol.* 15, 9, e1007276.

Plewczynski, Dariusz, Michal Laźniewski, Rafal Augustyniak, and Krzysztof Ginalski

2011 "Can We Trust Docking Results? Evaluation of Seven Commonly Used Programs on PDBbind Database", *J. Comput. Chem.* 32, 4, pp. 742-755.

Pouyanfar, Samira, Yudong Tao, Anup Mohan, Haiman Tian, Ahmed S Kaseb, Kent Gauen, Ryan Dailey, Sarah Aghajanzadeh, Yung-Hsiang Lu, Shu-Ching Chen, et al.

2018 "Dynamic Sampling in Convolutional Neural Networks for Imbalanced Data Classification", in *2018 IEEE Conf. on MIPR*, IEEE, pp. 112-117.

Prusis, Peteris, Ruta Muceniece, Per Andersson, Claes Post, Torbjörn Lundstedt, and Jarl E.S. Wikberg

2001 "PLS Modeling of Chimeric MS04/MSH-Peptide and MC1/MC3-Receptor Interactions Reveals a Novel Method for the Analysis of Ligand–Receptor interactions", *BBA - Protein Struct. M.* 1544, 1-2 (Jan. 2001), pp. 350-357.

Punta, Marco and Burkhard Rost

2008 "Neural Networks Predict Protein Structure and Function." *Methods Mol. Biol.* 458, pp. 203-230.

Putin, Evgeny, Arip Asadulaev, Yan Ivanenkov, Vladimir Aladinskiy, Benjamin Sanchez-Lengeling, Alán Aspuru-Guzik, and Alex Zhavoronkov

2018 "Reinforced Adversarial Neural Computer for de Novo Molecular Design", *J. Chem. Inf. Model.* 58, 6 (June 2018), pp. 1194-1204.

Qi, Yanjun, Merja Oja, Jason Weston, William Stafford Noble, and Z Zhang

2012 "A Unified Multitask Architecture for Predicting Local Protein Properties", *PLoS ONE*, 7, 3 (Mar. 2012), e32235.

Qian, Ning and Terrence J Sejnowski

1988 "Predicting the Secondary Structure of Globular Proteins Using Neural Network Models." *J. Mol. Biol.* 202, 4 (Aug. 1988), pp. 865-884.

Qiu, Tianyi, Jingxuan Qiu, Jun Feng, Dingfeng Wu, Yiyan Yang, Kailin Tang, Zhiwei Cao, and Ruixin Zhu

2017 "The Recent Progress in Proteochemometric Modelling: Focusing on Target Descriptors, Cross-Term Descriptors and Application Scope", *Brief. Bioinform.* 18, 1 (Jan. 2017), pp. 125-136.

Quang, Daniel and Xiaohui Xie

2016 "DanQ: a Hybrid Convolutional and Recurrent Deep Neural Network for Quantifying the Function of DNA Sequences", *Nucleic Acids Res.* 44, 11 (June 2016), e107-e107.

R Core Team

2015 *R: A Language and Environment for Statistical Computing*, R Foundation for Statistical Computing, Vienna, Austria, https://www.R-project.org/.

Radford, Alec, Luke Metz, and Soumith Chintala

2016 "Unsupervised Representation Learning With Deep Convolutional Generative Adversarial Networks", *arXiv*, 1511.06434v2.

Ragoza, Matthew, Joshua Hochuli, Elisa Idrobo, Jocelyn Sunseri, and David Ryan Koes

2017 "Protein - Ligand Scoring with Convolutional Neural Networks", *J. Chem. Inf. Model.* 57, pp. 942-957.

Ramsundar, Bharath

2016a *deepchem.io*, https://github.com/deepchem/deepchem.

2016b *deepchem.io*, https://github.com/deepchem/deepchem.

Ramsundar, Bharath, Steven Kearnes, Kearnes Edu, Patrick Riley, Dale Webster, David Konerding, and Vijay Pande

2015 "Massively Multitask Networks for Drug Discovery", *arXiv*, 1502.02072.

Ramsundar, Bharath, Bowen Liu, Zhenqin Wu, Andreas Verras, Matthew Tudor, Robert P Sheridan, and Vijay Pande

2017 "Is Multitask Deep Learning Practical for Pharma?", *J. Chem. Inf. Model.* 57, pp. 2068-2076.

Rao, Roshan, Nicholas Bhattacharya, Neil Thomas, Yan Duan, Xi Chen, John Canny, Pieter Abbeel, and Yun S. Song

2019 "Evaluating Protein Transfer Learning with TAPE", *arXiv*, 1906.08230.

Landrum, Greg

n.d. *RDKit: Open-Source Cheminformatics*, http://www.rdkit.org, (accessed May 25, 2018).

Reed, Scott, Zeynep Akata, Xinchen Yan, Lajanugen Logeswaran, Bernt Schiele, and Honglak Lee

2016 "Generative Adversarial Text to Image Synthesis", *arXiv*, 1605.05396v2.

Ren, Mengye, Wenyuan Zeng, Bin Yang, and Raquel Urtasun

2018 "Learning to Reweight Examples for Robust Deep Learning", in *Int. Conf. Mach. Learn.* PMLR, pp. 4334-4343.

Riesselman, Adam J., John B. Ingraham, and Debora S. Marks

2018 "Deep Generative Models of Genetic Variation Capture the Effects of Mutations", *Nat. Methods*, 15, 10, pp. 816-822.

Rifaioglu, Ahmet Sureyya, Tunca Dougan, Maria Jesus Martin, Rengul Cetin-Atalay, and Volkan Atalay

2019 "DEEPred: Automated Protein Function Prediction with Multi-task Feed-forward Deep Neural Networks", *Sci. Rep.* 9, 1, p. 7344.

Riniker, Sereina and Gregory A. Landrum

2013 "Open-Source Platform to Benchmark Fingerprints for Ligand-Based Virtual Screening", *J. Cheminf.* 5, 1 (May 2013), p. 26.

Rives, Alexander, Goyal Siddharth, Joshua Meier, Demi Guo, Myle Ott, C. Lawrence Zitnick, Jerry Ma, and Rob Fergus

2019 "Biological Structure and Function Emerge from Scaling UNsupervised Learning to 250 Million Protein Sequences", *bioRxiv*.

Rodriguez-Perez, Raquel and Jürgen Bajorath

2019 "Multitask Machine Learning for Classifying Highly and Weakly Potent Kinase Inhibitors", *ACS Omega*, 4, 2 (Feb. 2019), pp. 4367-4375.

Rogers, David and Mathew Hahn

2010 "Extended-Connectivity Fingerprints", *J. Chem. Inf. Model*, 50, 5 (May 2010), pp. 742-754.

Rognan, Didier

2007 "Chemogenomic Approaches to Rational Drug Design." *Brit. J. Pharmacol.* 152, 1 (Sept. 2007), pp. 38-52.

Rohrer, Sebastian G. and Knut Baumann

2009 "Maximum Unbiased Validation (MUV) Data Sets for Virtual Screening Based on PubChem Bioactivity Data", *J. Chem. Inf. Model.* 49, 2 (Feb. 2009), pp. 169-184.

Rosenblatt, Frank

1958 "The Perceptron: a Probabilistic Model for Information Storage and Organization in the Brain." *Psychol. Rev.* 65, 6, p. 386.

Ross, T-YLPG and GKHP Dollár

2017 "Focal Loss for Dense Object Detection", in *Proc. of the IEEE CCVP*, pp. 2980-2988.

Rudin, Cynthia

2019 "Stop Explaining Black Box Machine Learning Models for High Stakes Decisions and Use Interpretable Models Instead", *Nat. Mach. Intell.* 1, pp. 206-215.

Ruiz-Carmona, Sergio, Daniel Alvarez-Garcia, Nicolas Foloppe, A. Beatriz Garmendia-Doval, Szilveszter Juhos, Peter Schmidtke, Xavier Barril, Roderick E. Hubbard, and S. David Morley

2014 "rDock: A Fast, Versatile and Open Source Program for Docking Ligands to Proteins and Nucleic Acids", *PLoS Comput. Biol.* 10, 4 (Apr. 2014), e1003571.

Rumelhart, David E., Geoffrey E. Hinton, and Ronald J. Williams

1986 "Learning Representations by Back-Propagating Errors", *Nature*, 323, 6088 (Oct. 1986), pp. 533-536.

Salentin, Sebastian, Sven Schreiber, V Joachim Haupt, Melissa F Adasme, and Michael Schroeder

2015 "PLIP: Fully Automated Protein–Ligand Interaction Profiler", *Nucleic Acids Res.* 43, W1, W443-W447.

Salomon-Ferrer, Romelia, David A. Case, and Ross C. Walker

2013 "An Overview of the Amber Biomolecular Simulation Package", *WIRES Comput. Mol. Sci.* 3, 2, pp. 198-210.

Salsbury, Freddie R and Jr.

2010 "Molecular Dynamics Simulations of Protein Dynamics and their Relevance to Drug Discovery." *Curr. Opin. Pharmacol.* 10, 6 (Dec. 2010), pp. 738-44.

Sanchez-Lengeling, Benjamin, Carlos Outeiral, Gabriel L Guimaraes, and Alán Aspuru-Guzik

2017 "Optimizing Distributions over Molecular Space. An Objective-Reinforced Generative Adversarial Network for Inverse-design Chemistry (ORGANIC)", *ChemRxiv*, 5309668.v3, pp. 1-18.

Sastry, Madhavi, Jeffrey F. Lowrie, Steven L. Dixon, and Woody Sherman

2010 "Large-Scale Systematic Analysis of 2D Fingerprint Methods and Parameters to Improve Virtual Screening Enrichments", *J. Chem. Inf. Model.* 50, 5 (May 2010), pp. 771-784.

Sawada, Ryusuke, Masaaki Kotera, and Yoshihiro Yamanishi

2014 "Benchmarking a Wide Range of Chemical Descriptors for Drug-Target Interaction Prediction Using a Chemogenomic Approach", *Mol. Inform.* 33, 11-12, pp. 719-731.

Schuster, Mike and Kuldip K Paliwal

1997 "Bidirectional Recurrent Neural Networks", *IEEE T. Signal Proces.* 45, 11.

Shen, Hong Bin and Kuo Chen Chou

2007 "EzyPred: A Top-Down Approach for Predicting Enzyme Functional Classes and Subclasses", *Biochem. Biophys. Res. Commun.* 364, 1, pp. 53-59.

Shen, Hong-Bin and Kuo-Chen Chou

2008 "PseAAC: a Flexible Web Server for Generating Various Kinds of Protein Pseudo Amino Acid Composition", *Anal. Biochem.* 373, 2, pp. 386-388.

Shen, Min, Cécile Béguin, Alexander Golbraikh, James P. Stables, Harold Kohn, and Alexander Tropsha

2004 "Application of Predictive QSAR Models to Database Mining: Identification and Experimental Validation of Novel Anticonvulsant Compounds", *J. Med. Chem.* 47, 9 (Apr. 2004), pp. 2356-2364.

Shen, Zhen, Wenzheng Bao, and De-Shuang Huang

2018 "Recurrent Neural Network for Predicting Transcription Factor Binding Sites", *Sci. Rep.* 8, 1, pp. 1-10.

Sheridan, Robert P.

2013 "Time-Split Cross-Validation as a Method for Estimating the Goodness of Prospective Prediction." *J. Chem. Inf. Model.* 53, 4 (Apr. 2013), pp. 783-790.

Sheridan, Robert P., Wei Min Wang, Andy Liaw, Junshui Ma, and Eric M. Gifford

2016 "Extreme Gradient Boosting as a Method for Quantitative Structure–Activity Relationships", *J. Chem. Inf. Model.* 56, 12 (Dec. 2016), pp. 2353-2360.

Shorten, Connor and Taghi M. Khoshgoftaar

2019 "A Survey on Image Data Augmentation for Deep Learning", *J. Big Data*, 6, 1 (Dec. 2019).

Shrikumar, Avanti, Peyton Greenside, and Anshul Kundaje

2017 "Learning Important Features Through Propagating Activation Differences", *arXiv*, 1704.02685.

Shrikumar, Avanti, Katherine Tian, Žiga Avsec, Anna Shcherbina, Abhimanyu Banerjee, Mahfuza Sharmin, Surag Nair, and Anshul Kundaje

2018 "Technical Note on Transcription Factor Motif Discovery from Importance Scores (TF-MoDISco) version 0.5.1.1", *arXiv*, 1811.00416.

Simonyan, Karen, Andrea Vedaldi, and Andrew Zisserman

2014 "Deep Inside Convolutional Networks : Visualising Image Classification Models and Saliency Maps", *arXiv*, 1312.6034v2.

Skiena, Steven S

2017 *The Data Science Design Manual*, Springer.

Soltanayev, Shakarim and Se Young Chun

2018 "Training and Refining Deep Learning Based Denoisers without Ground Truth Data", *arXiv*, 1803.01314.

Sönderby, Sören Kaae, Casper Kaae Sönderby, Henrik Nielsen, and Ole Winther
2015 "Convolutional LSTM Networks for Subcellular Localization of Proteins", *arXiv*, 1503.01919v1.

Song, Congzheng and Ananth Raghunathan
2020 "Information Leakage in Embedding Models", *arXiv*, 2004.00053.

Spencer, Matt, Jesse Eickholt, and Jianlin Jianlin Cheng
2015 "A Deep Learning Network Approach to ab initio Protein Secondary Structure Prediction." *IEEE ACM T. Comput. Bi.* 12, 1, pp. 103-112.

Srivastava, Nitish, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov
2014 "Dropout: A Simple Way to Prevent Neural Networks from Overfitting", *J. Mach. Learn. Res.* 15, pp. 1929-1958.

Srivastava, Santosh Kumar, Vinay Kumar Khanna, Shikha Gupta, Feroz Khan, and Dharmendra K. Yadav
2012 "Method for Predicting and Modeling Anti-Psychotic Activity Using Virtual Screening Model", WO 2012042541A2.

Stepniewska-Dziubinska, Marta M., Piotr Zielenkiewicz, and Pawel Siedlecki
2018 "Development and Evaluation of a Deep Learning Model for Protein-Ligand Binding Affinity Prediction", *Bioinformatics*, 34, 21, pp. 3666-3674.

Sterling, Teague and John J. Irwin
2015 "ZINC 15 - Ligand Discovery for Everyone", *J. Chem. Inf. Model.* 55, 11, pp. 2324-2337.

Strömbergsson, Helena, Maris Lapins, Gerard J. Kleywegt, and Jarl E. S. Wikberg
2010 "Towards Proteome-Wide Interaction Models Using the Proteochemometrics Approach", *Mol. Inform.* 29, 6-7 (June 2010), pp. 499-508.

Surgand, Jean-Sebastien, Jordi Rodrigo, Esther Kellenberger, and Didier Rognan
2005 "A Chemogenomic Analysis of the Transmembrane Binding Cavity of Human G-Protein-Coupled Receptors", *Proteins*, 62, 2 (Nov. 2005), pp. 509-538.

Suzek, Baris E., Yuqi Wang, Hongzhan Huang, Peter B. McGarvey, Cathy H. Wu, and the UniProt Consortium
2014 "UniRef Clusters: A Comprehensive and Scalable Alternative for Improving Sequence Similarity Searches", *Bioinformatics*, 31, 6 (Nov. 2014), pp. 926-932, ISSN: 1367-4803.

Svetnik, Vladimir, Ting Wang, Christopher Tong, Andy Liaw, Robert P. Sheridan, and Qinghua Song
2005 "Boosting: An Ensemble Learning Tool for Compound Classification and QSAR Modeling", *J. Chem. Inf. Model.* 45, 3 (May 2005), pp. 786-799.

Szalkai, Balazs and Vince Grolmusz

2018 "SECLAF: A Webserver and Deep Neural Network Design Tool for Hierarchical Biological Sequence Classification", *Bioinformatics*, 34, 14, pp. 2487-2489.

Tammiku-Taul, Jaana, Rahel Park, Kaur Jaanson, Kristi Luberg, Dimitar A Dobchev, Dzmitry Kananovich, Artur Noole, Merle Mandel, Allen Kaasik, Margus Lopp, et al.

2016 "Indole-Like Trk Receptor Antagonists", *Eur. J. Med. Chem.* 121, pp. 541-552.

Tang, Hao, Xiang S. Wang, Xi-Ping Huang, Bryan L. Roth, Kyle V. Butler, Alan P. Kozikowski, Mira Jung, and Alexander Tropsha

2009 "Novel Inhibitors of Human Histone Deacetylase (HDAC) Identified by QSAR Modeling of Known Inhibitors, Virtual Screening, and Experimental Validation", *J. Chem. Inf. Model.* 49, 2 (Feb. 2009), pp. 461-476.

The Association of the British Pharmaceutical Industry

2018 *From Molecule to Medicine*, https://www.abpi.org.uk/media-centre/blog/2018/august/from-molecule-to-medicine/.

Theano Development Team

2016 "Theano: A Python Framework for Fast Computation of Mathematical Expressions", *arXiv*, 1605.02688 (May 2016).

Tian, Kai, Mingyu Shao, Yang Wang, Jihong Guan, and Shuigeng Zhou

2016 "Boosting Compound-Protein Interaction Prediction by Deep Learning", *Methods*, 110, pp. 64-72.

Todeschini, Roberto and Viviana Consonni

2008 *Handbook of Molecular Descriptors*, 1st ed., Wiley-VCH, New York, vol. 11, p. 667, ISBN: 3527613110.

Torng, Wen and Russ B Altman

2017 "3D Deep Convolutional Neural Networks for Amino Acid Environment Similarity Analysis", *BMC Bioinf.* 18, 1, p. 302.

Trabelsi, Ameni, Mohamed Chaabane, and Asa Ben-Hur

2019 "Comprehensive Evaluation of Deep Learning Architectures for Prediction of DNA/RNA Sequence Binding Specificities", *Bioinformatics*, 35, 14, pp. i269-i277.

Tripepi, G., K. J. Jager, F. W. Dekker, C. Wanner, and C. Zoccali

2008 "Bias in Clinical Research", *Kidney Int.* 73, 2 (Jan. 2008), pp. 148-153.

Trott, Oleg and Arthur J Olson

2010 "AutoDock Vina: Improving the Speed and Accuracy of Docking with a New Scoring Function, Efficient Optimization, and Multithreading." *J. Comput. Chem.* 31, 2 (Jan. 2010), pp. 455-61.

Truchon, Jean-Franois, Christopher I Bayly, and Jean-François Truchon

2007 "Evaluating Virtual Screening Methods: Good and Bad Metrics for the " Early Recognition " Problem", *J. Chem. Inf. Model*, 47, 2, pp. 488-508.

UniProt, Consortium

2017 "UniProt: the Universal Protein Knowledgebase", *Nucleic Acids Res.* 45, D1 (Jan. 2017), pp. D158-D169.

Unterthiner, Thomas, Andreas Mayr, Günter Klambauer, Marvin Steijaert, Jörg K Wegner, Hugo Ceulemans, and Sepp Hochreiter

2014 "Deep Learning for Drug Target Prediction", in *Conf. Neur. Inf. Proc. Sys. Found. (NIPS 2014), Workshop Rep. Learn. Complex Out.*

van Westen, Gerard J P, Jörg K. Wegner, Adriaan P. Ijzerman, Herman W T van Vlijmen, and A. Bender

2011 "Proteochemometric Modeling as a Tool to Design Selective Compounds and for Extrapolating to Novel Targets", *Med. Chem. Comm.* 2, 1, pp. 16-30.

Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin

2017 "Attention Is All You Need", *arXiv*, 1706.03762v4.

Veber, Daniel F, Stephen R Johnson, Hung-yuan Cheng, Brian R Smith, Keith W Ward, and Kenneth D Kopple

2002 "Molecular Properties That Influence the Oral Bioavailability of Drug Candidates", *J. Med. Chem.* 45, pp. 2615-2623.

Verdonk, Marcel L., Jason C. Cole, Michael J. Hartshorn, Christopher W. Murray, and Richard D. Taylor

2003 "Improved Protein-Ligand Docking Using GOLD", *Proteins*, 52, 4 (Aug. 2003), pp. 609-623.

Vidal, David, Michael Thormann, and Miquel Pons

2005 "LINGO, an Efficient Holographic Text Based Method to Calculate Biophysical Properties and Intermolecular Similarities", *J. Chem. Inf. Model*, 45, 2, pp. 386-393.

Vincent, Pascal, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol

2008 "Extracting and Composing Robust Features with Denoising Autoencoders", *Proc. 25th Int.Conf. Mach. Learn.* Pp. 1096-1103.

Vinyals, Oriol, Samy Bengio, and Manjunath Kudlur

2016 "Order Matters: Sequence to Sequence for Sets", *4th ICLR - Conf. Track Proc.*

Wallach, Izhar, Michael Dzamba, and Abraham Heifets

2015 "AtomNet: A Deep Convolutional Neural Network for Bioactivity Prediction in Structure-based Drug Discovery", *arXiv*, 1510.02855 (Oct. 2015).

Wallach, Izhar and Abraham Heifets

2018 "Most Ligand-Based Classification Benchmarks Reward Memorization Rather than Generalization", *J. Chem. Inf. Model.* 58, 5 (May 2018), pp. 916-932.

Walter, S. D.

2005 "The Partial Area Under the Summary ROC Curve", *Stat. Med.* 24, 13 (July 2005), pp. 2025-2040.

Wan, Fangping and Jianyang Zeng

2016 "Deep Learning with Feature Embedding for Compound-Protein Interaction Prediction", *bioRxiv*.

Wang, Haishuai, Zhicheng Cui, Yixin Chen, Michael Avidan, Arbi Ben Abdallah, and Alexander Kronzer

2018 "Predicting Hospital Readmission via Cost-Sensitive Deep Learning", *IEEE/ACM Trans. Comput. Biol. Bioinf.* 15, 6, pp. 1968-1978.

Wang, Jiawei, Bingjiao Yang, Jerico Revote, Andre Leier, Tatiana T Marquez-Lago, Geoffrey Webb, Jiangning Song, Kuo-Chen Chou, and Trevor Lithgow

2017 "POSSUM: a Bioinformatics Toolkit for Generating Numerical Sequence Feature Descriptors Based on PSSM Profiles", *Bioinformatics*, 33, 17 (May 2017), pp. 2756-2758.

Wang, Liangjiang, Caiyan Huang, Mary Qu Yang, and Jack Y Yang

2010 "BindN+ for Accurate Prediction of DNA and RNA-binding Residues from Protein Sequence Features", *BMC Syst. Biol.* 4, S1, S3.

Wang, Sheng, Yuzhi Guo, Yuhong Wang, Hongmao Sun, and Junzhou Huang

2019 "SMILES-BERT: Large Scale Unsupervised Pre-Training for Molecular Property Prediction", *Proc. 10th Int. Conf. ACM-BCB*, pp. 429-436.

Wang, Shoujin, Wei Liu, Jia Wu, Longbing Cao, Qinxue Meng, and Paul J Kennedy

2016 "Training Deep Neural Networks on Imbalanced Data Sets", in *2016 IJCNN*, IEEE, pp. 4368-4374.

Wang, Yan Bin, Zhu Hong You, Shan Yang, Hai Cheng Yi, Zhan Heng Chen, and Kai Zheng

2020 "A Deep Learning-based Method for Drug-Target Interaction Prediction Based on Long Short-Term Memory Neural Network", *BMC Med. Inform. Decis.* 20, Suppl 2, pp. 1-9.

Wang, Yanli, Stephen H. Bryant, Tiejun Cheng, Jiyao Wang, Asta Gindulyte, Benjamin A. Shoemaker, Paul A. Thiessen, Siqian He, and Jian Zhang

2017 "PubChem BioAssay: 2017 Update", *Nucleic Acids Res.* 45, D1, pp. D955-D963.

Wang, Yanli, Jewen Xiao, Tugba O Suzek, Jian Zhang, Jiyao Wang, Zhigang Zhou, Lianyi Han, Karen Karapetyan, Svetlana Dracheva, Benjamin A Shoemaker, Evan Bolton, Asta Gindulyte, and Stephen H Bryant

2012 "PubChem's BioAssay Database", *Nucleic Acids Res.* 40, Database issue (Jan. 2012), pp. D400-12.

Weininger, David

1988 "SMILES, a Chemical Language and Information System. 1. Introduction to Methodology and Encoding Rules", *J. Chem. Inf. Model.* 28, 1 (Feb. 1988), pp. 31-36.

Wen, Ming, Zhimin Zhang, Shaoyu Niu, Haozhi Sha, Ruihan Yang, Yonghuan Yun, and Hongmei Lu

2017 "Deep-Learning-Based Drug-Target Interaction Prediction", *J. Proteome Res.* 16, 4, pp. 1401-1409.

Wilcoxon, Frank

1945 *Individual Comparisons by Ranking Methods*, tech. rep. 6, pp. 80-83.

Williams, Ronald J and David Zipser

1989 "A Learning Algorithm for Continually Running Fully Recurrent Neural Networks", *Neural Comput.* 1, 2, pp. 270-280.

Wishart, David S, Yannick D Feunang, An C Guo, Elvis J Lo, Ana Marcu, Jason R Grant, Tanvir Sajed, Daniel Johnson, Carin Li, Zinat Sayeeda, et al.

2018 "DrugBank 5.0: a Major Update to the DrugBank Database for 2018", *Nucleic Acids Res.* 46, D1, pp. D1074-D1082.

Wu, Dingfeng, Qi Huang, Yida Zhang, Qingchen Zhang, Qi Liu, Jun Gao, Zhiwei Cao, and Ruixin Zhu

2012 "Screening of Selective Histone Deacetylase Inhibitors by Proteochemometric Modeling", *BMC Bioinformatics*, 13, 1, pp. 0–9.

Wu, Zhenqin, Bharath Ramsundar, Evan N Feinberg, Joseph Gomes, Caleb Geniesse, Aneesh S Pappu, Karl Leswing, and Vijay Pande

2017 "MoleculeNet: A Benchmark for Molecular Machine Learning", *arXiv*, 1703.00564.

Xia, Jie, Hongwei Jin, Zhenming Liu, Liangren Zhang, and Xiang Simon Wang

2014 "An Unbiased Method To Build Benchmarking Sets for Ligand-Based Virtual Screening and its Application To GPCRs", *J. Chem. Inf. Model.* 54, 5 (May 2014), pp. 1433-1450.

Xu, Kelvin, Lei Ba Jimmy, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhutdinov, Richard S Zemel, and Yoshua Bengio

2016 "Show, Attend and Tell: Neural Image Caption Generation with Visual Attention", *arXiv*, 1502.03044v3.

Xu, Youjun, Ziwei Dai, Fangjin Chen, Shuaishi Gao, Jianfeng Pei, and Luhua Lai

2015 "Deep Learning for Drug-Induced Liver Injury", *J. Chem. Inf. Model.* 55, 10 (Oct. 2015), pp. 2085-2093.

Yan, ShiDu and Koteswara Rao Valasani

2017 "Phosphonate Derivatives and Methods of Use thereof in the Treatment of Alzheimer's Disease", US 9738668B2.

Yang, Kevin K., Zachary Wu, Claire N. Bedbrook, and Frances H. Arnold

2018 "Learned Protein Embeddings for Machine Learning", *Bioinformatics*, 34, 15, pp. 2642-2648.

Yang, Xin, Yifei Wang, Ryan Byrne, Gisbert Schneider, and Shengyong Yang

2019 "Concepts of Artificial Intelligence for Computer-Assisted Drug Discovery", *Chem. Rev.* 119, 18, pp. 10520-10594.

Yosinski, Jason, Jeff Clune, and Thomas Fuchs

2015 "Understanding Neural Networks Through Deep Visualization", *arXiv*, 1506.06579v1.

Young, Tom, Devamanyu Hazarika, Soujanya Poria, and Erik Cambria

2018 "Recent Trends in Deep Learning Based Natural Language Processing", *arXiv*, 1708.02709.

Yu, Dong, Adam Eversole, Mike Seltzer, Kaisheng Yao, Oleksii Kuchaiev, Yu Zhang, Frank Seide, Zhiheng Huang, Brian Guenter, Huaming Wang, Jasha Droppo, Geoffrey Zweig, Chris Rossbach, Jie Gao, Andreas Stolcke, Jon Currey, Malcolm Slaney, Guoguo Chen, Amit Agarwal, Chris Basoglu, Marko Padmilac, Alexey Kamenev, Vladimir Ivanov, Scott Cypher, Hari Parthasarathi, Bhaskar Mitra, Baolin Peng, and Xuedong Huang

2014 *An Introduction to Computational Networks and the Computational Network Toolkit*, tech. rep.

Zakharov, Alexey V, Megan L Peach, Markus Sitzmann, and Marc C Nicklaus

2014 "QSAR Modeling of Imbalanced High-Throughput Screening Data in PubChem", *J. Chem. Inf. Model.* 54, 3, pp. 705-712.

Zakharov, Alexey V, Tongan Zhao, Dac-Trung Nguyen, Tyler Peryea, Timothy Sheils, Adam Yasgar, Ruili Huang, Noel Southall, and Anton Simeonov

2019 "Novel consensus architecture to improve performance of large-scale multitask deep learning QSAR models", *J. Chem. Inf. Model.* 59, 11, pp. 4613-4624.

Zeiler, Matthew D and Rob Fergus

2014 "Visualizing and Understanding Convolutional Networks", in *Computer Vision – ECCV 2014*, ed. by David Fleet, Tomas Pajdla, Bernt Schiele, and Tinne Tuytelaars, Springer International Publishing, Cham, pp. 818-833, ISBN: 978-3-319-10590-1.

Zhang, Bijun, Martin Vogt, Gerald M. Maggiora, and Jürgen Bajorath

2015 "Design of Chemical Space Networks Using a Tanimoto Similarity Variant Based upon Maximum Common Substructures", *J. Comput. Aided Mol. Des.* 29, 10 (Oct. 2015), pp. 937-950.

Zhang, Chong, Kay Chen Tan, and Ruoxu Ren

2016 "Training Cost-Sensitive Deep Belief Networks on Imbalance Data Problems", in *2016 IJCNN*, IEEE, pp. 4362-4367.

Zhang, P, L Tao, X Zeng, C Qin, SY Chen, F Zhu, SY Yang, ZR Li, WP Chen, and YZ Chen

2017 "PROFEAT Update: a Protein Features Web Server with Added Facility to Compute Network Descriptors for Studying Omics-Derived Networks", *J. Mol. Biol.* 429, 3, pp. 416-425.

Zhang, Shuxing, Linyi Wei, Ken Bastow, Weifan Zheng, Arnold Brossi, Kuo-Hsiung Lee, and Alexander Tropsha

2007 "Antitumor Agents 252. Application of Validated QSAR Models to Database Mining: Discovery of Novel Tylophorine Derivatives as Potential Anticancer Agents", *J. Comput. Aid. Mol. Des.* 21, 1-3 (Mar. 2007), pp. 97-112.

Zheng, Xueming, Shungao Xu, Ying Zhang, and Xinxiang Huang

2019 "Nucleotide-Level Convolutional Neural Networks for pre-miRNA Classification", *Sci. Rep.* 9, 1, p. 628.

Zhou, Xuan, Zhanchao Li, Zong Dai, and Xiaoyong Zou

2011 "Predicting Methylation Status of Human DNA Sequences by Pseudo-Trinucleotide Composition", *Talanta*, 85, 2, pp. 1143-1147.