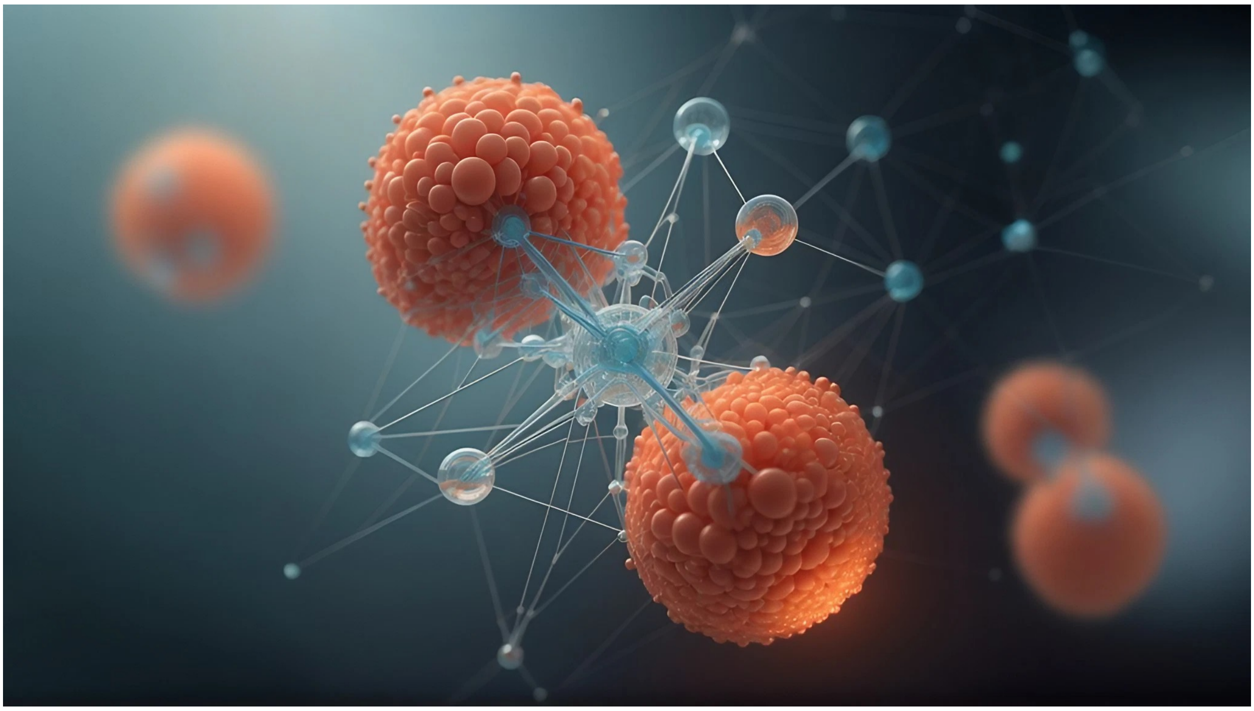
CHALMERS
UNIVERSITY OF TECHNOLOGY | UNIVERSITY OF GOTHENBURG



Machine Learning for Predicting Targeted Protein Degradation

A deep learning-based system for predicting protein degradation activity in PROTAC systems

Master's thesis in Computer Science and Engineering

STEFANO RIBES

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2023

Master's thesis 2023

Machine Learning for Predicting Targeted Protein Degradation

A deep learning-based system for predicting protein degradation activity in PROTAC systems

STEFANO RIBES

UNIVERSITY OF GOTHENBURG

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Computer Science and Engineering
Chalmers University of Technology
University of Gothenburg
Gothenburg, Sweden 2023

Machine Learning for Predicting Targeted Protein Degradation
A deep learning-based system for predicting protein degradation activity in PROTAC systems
STEFANO RIBES

Supervisor: Rocío Mercado, CSE Department at Chalmers
Advisor: Eva Nittinger, AstraZeneca
Advisor: Christian Tyrchan, AstraZeneca
Examiner: Peter Damaschke, CSE Department at Chalmers

Master's Thesis 2023
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: I generated the image in Stable Diffusion, with the prompt: "Two proteins linked together by atoms. Neural network in the background. Chemistry, bioinformatics, computer science".

Typeset in LaTeX
Gothenburg, Sweden 2023

Machine Learning for Predicting Targeted Protein Degradation
A deep learning-based system for predicting protein degradation activity in PROTAC systems
STEFANO RIBES
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg

Abstract

PROteolysis TArgeting Chimeras (PROTACs) are an emerging high-potential therapeutic technology. PROTACs leverage the ubiquitination and proteasome processes within a cell to degrade a Protein Of Interest (POI). Designing new PROTAC molecules, however, is a challenging task, as assessing the degradation efficacy of PROTACs often requires extensive effort, mostly in terms of expertise, cost and time, for instance via laboratory assays. Machine Learning (ML) and Deep Learning (DL) technologies are revolutionizing many scientific fields, including the drug development pipeline. In this thesis, we present the data collection and curation strategy, as well as several candidate DL models, for ultimately predicting the degradation efficacy of PROTAC molecules. In order to train and evaluate our system, we propose a curated version of open source datasets from literature. Relevant features such as pDC_{50}, D_{max}, E3 ligase type, POI amino acid sequence, and experimental cell type are carefully organized and parsed via a Named Entity Recognition system based on a BERT model. The curated datasets have been used for developing three candidate DL models. Each DL model is designed to leverage different PROTAC representations: molecular fingerprints, molecular graphs and tokenized SMILES. The proposed models are evaluated against an XGBoost model baseline and the State-of-The-Art (SOTA) model for predicting PROTACs degradation activity. Overall, our best DL models achieved a validation accuracy of 80.26% versus SOTA's 77.95% score, and a Receiver Operating Characteristic Area Under the Curve (ROC AUC) validation score of 0.849 versus SOTA' 0.847.

Keywords: Deep learning, Chemoinformatics, PROTAC, Drug design.

Acknowledgements

I would like to express my deepest gratitude to the people who have contributed to the successful completion of my master's thesis. Their guidance, support, and encouragement have been invaluable throughout this journey.

First and foremost, I would like to thank my supervisors Rocío, Eva and Christian. Your expertise, guidance, and patience have been invaluable throughout this research journey. Your practical insights, industry knowledge, and willingness to share your expertise have provided a crucial real-world perspective to my work. I would like to give a word of appreciation to my examiner Peter Damaschke, who always ensured high standards for this thesis work.

Grazie a voi, mamma, papá e Fil, che nonostante la distanza mi avete sempre incoraggiato e creduto in me. Grazie Fil per tutte le volte che ci siamo confrontati e per la pazienza con cui mi hai sempre ascoltato.

I would like to express my heartfelt appreciation to my girlfriend, Marghe. Your love, patience, and unwavering support have been my anchor throughout this challenging endeavor. Your presence by my side, cheering me on and celebrating every milestone, has made this journey all the more meaningful. I am truly fortunate to have you in my life.

It might be impossible to mention everyone who, in these months, help me or with whom I simply shared valuable moments together, but I can try... So, thanks Kinga, Giulia, Mert, Gustav, Dani, Maciej, Lin, Emma, Leonardo, Hongtao, Martin Raum, Mors, Corra, Los, Tudor, Matte, Luca, Foro, Sanchi, Sofi, and thanks to everyone reading this thesis and thinking "Ehi I know this guy, why am I not here too?" (both my memory and this page have a limited space, sorry sorry)

Finally, I would like to say that working on this thesis has reignited my passion for research. The process of delving into the subject matter, analyzing data, and conducting experiments, has reminded me of the joy and fulfillment that research brings. To all the people that made this thesis possible, I am grateful for this opportunity and I am excited about the possibilities that lie ahead.

Stefano, Gothenburg, 2023-07-06

Contents

List of Figures

List of Tables

Acronyms

DC_{50} Half maximal degradation concentration. 4

D_{max} Maximal effect. 4

IC_{50} Half-maximal Inhibitory Concentration. 13

AI Artificial Intelligence. 8

BERT Bidirectional Encoder Representations from Transformers. 11

DL Deep Learning. 1, 8

DL Machine Learning. 1

GAT Graph ATtention. 9

GC Graph Convolution. 9

GNN Graph Neural Networks. 9

MACCS Molecular ACCess System. 6

MLM Masked Language Model. 19

MLP Multi-Layer Perceptron. 9

NER Named Entity Recognition. 15

NLP Natural Language Processing. 8

PLI Protein-Ligand Interactions. 1

POI Protein Of Interest. 1

PROTAC PROteolysis TArgeting Chimeras. 1

ReLU Rectified Linear Unit. 9

RMSE Root Mean Squared Error. 13

ROC AUC Receiver Operating Characteristic Area Under the Curve. v, 27

SMILES Simplified Molecular-Input Line-Entry System. 7

SSL Self-Supervised Learning. 18

1

Introduction

Accurate prediction of Protein-Ligand Interactions (PLI) is an essential step in therapeutics design and discovery. One area in which PLIs are particularly challenging to predict is in the design of PROteolysis TArgeting Chimeras (PROTAC) [1]. PROTACs are an interesting emerging modality in medicinal chemistry. PROTACs bind to a Protein Of Interest (POI) implicated in a particular disease pathway and at the same time to a ligase, such as an E3 ubiquitin ligase. Bringing these into close contact leads to ubiquitination of the POI and its eventual degradation by the proteasome, effectively destroying the POI and making it unable to fulfil its function anymore.

PROTACs offer several advantages in drug development [1]. PROTACs have the potential for low doses due to their ability to degrade multiple target proteins, enabling effective treatment with reduced systemic toxicity. Additionally, PROTACs can leverage even weak binding interactions with target proteins, expanding the range of druggable targets. This unique feature enables the development of therapeutics for proteins that lack well-defined binding sites, greatly broadening the scope of drug discovery.[1] Many studies are also showing the potential of PROTACs in the treatment of diseases, such as neurodegenerative disorders [2] or cancers [3], by targeting specific proteins associated with the pathogenesis of these diseases. However, as experience with PROTACs has grown, limitations have also emerged, such as PROTACs' large sizes, limited cell permeabilities, and limited number of suitable E3 ligases to exploit for ubiquitination [1]. In particular, PROTACs do not easily degrade many types of proteins and their performance depends on several diverse factors, such as the binding affinity of the PROTAC and the POI, which is often hard to predict *in silico*, *i.e.*, computationally. In fact, the performance of PROTACs is typically assessed using a specifically designed concentration-response assay, a laboratory test that measures the activity of the target protein (*i.e.*, the POI) in the presence of different concentrations of the candidate PROTAC. Nonetheless, designing and conducting such assays is generally a complex, time consuming, and expensive procedure.

For all these reasons, the use of Machine Learning (DL) models, and in particular Deep Learning (DL) models, could speed up the evaluation and identification of novel

[1] Conventional drugs, on the other hand, typically work by directly binding to specific target proteins and modulating their activity. They often act as inhibitors or activators, interfering with the normal function of the target protein. This modulation can alter signaling pathways, block the interaction with other molecules, or inhibit enzymatic activity, resulting in a desired therapeutic effect.

PROTAC candidates. In the ideal scenario, having a system for accurately predicting PROTAC performance would limit the number of experiments needed to identify the most promising PROTACs in a drug discovery campaign, effectively accelerating the overall drug design process. On top of that, accurate predictions of PROTAC efficacy can be used as a scoring metric to boost the performance of molecular deep generative models, like reinforcement learning-based [4] or diffusion-based generative models, and in turn improve the *de novo* design of PROTAC molecules.

In this project we developed a machine learning system for predicting PROTAC degradation activity in terms of DC_{50} and D_{max}. To do so, we propose, design, and evaluate several candidate models for predicting the capacity for protein degradation of a given PROTAC. To train and assess our system, we introduce a curated version of the open-source PROTAC-DB [5] and PROTAC-Pedia [6] datasets. We compare our proposed models with an XGBoost model baseline and the recent DeepPROTACs model [7]. Our top-performing deep learning models achieved a validation accuracy of 80.26%, outperforming DeepPROTACs' score of 77.95%, and attained an AUC of 0.849 compared to DeepPROTACs' 0.847.

The remainder of this report is organized as follows. Chapter 2 includes background information for better understanding our research work. Chapter 3 gives an overview of the literature work related to this thesis. Then, Chapter 4 builds upon the background theory to detail our followed methodology. Finally, in Chapter 5 we show and discuss the obtained results before drawing our conclusions in Chapter 6.

2

Background

This chapter introduces and gives a general overview of the main theoretical concepts behind protein degradation via PROTACs and machine learning algorithms, with a focus on deep learning applied to the drug design process.

2.1 Protein Degradation and PROTACs

Proteins are large biomolecules made of long chains of amino acids and are responsible for virtually any function within living beings. As proteins are constantly generated inside a cell, cells regulate the quantity of specific proteins by getting rid of those which are unnecessary, do not fold correctly, or do not function properly. One way to degrade and recycle such unnecessary proteins is via a process involving the *ubitiquination* and *proteolysis* mechanisms, depicted in the upper part of Figure 2.1. During ubitiquination, in the top-left corner, a E3 ligase protein binds to the target and recruits another ligase, E2, which carries an ubiquitin protein and transfers it to certain amino acids on the surface of the target protein. The target protein gets ubiquitinated and is thus marked for degradation. In the end, the degradation, also called *proteolysis*, of a target is performed by the *proteasome*, a protein which breaks ubiquitin-marked peptide bonds of the target protein.

PROTACs are bifunctional small molecules[1] that leverage the aforementioned intracellular degradation process in order to degrade a target POI (*e.g.*, a foreign protein originated from a viral infection). To do so, PROTACs consist of three main components, highlighted in the bottom left of Figure 2.1: a *warhead* which binds to a POI, a *E3 ligand* which binds to a specific E3 ligase protein, and a *linker* which connects the warhead and the ligand together, *i.e.*, the two binding domains. Thanks to these different elements, PROTACs can then bind to a POI and to the E3 ligase, bringing them close together. If a successful ternary structure, also called *ternary complex*, is formed, the E3 ligase will then ubiquitinate the POI, eventually triggering the POI's degradation by the proteasome, as showed in the bottom right part of Figure 2.1. In this scenario, PROTACs represent an elegant solution for "connecting" a E3 ligase enzyme to ideally any POI, even when the E3 does not bind directly to the POI.

[1] PROTACs are classified as small molecules, however, they are rather large since they consist of two small molecules connected via a linker.

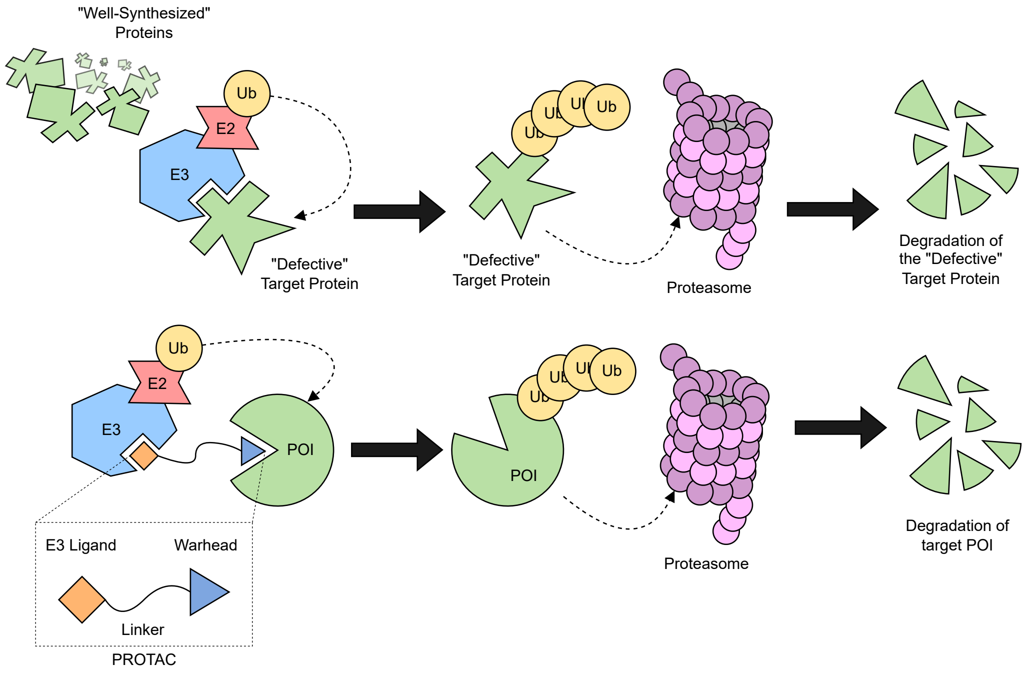


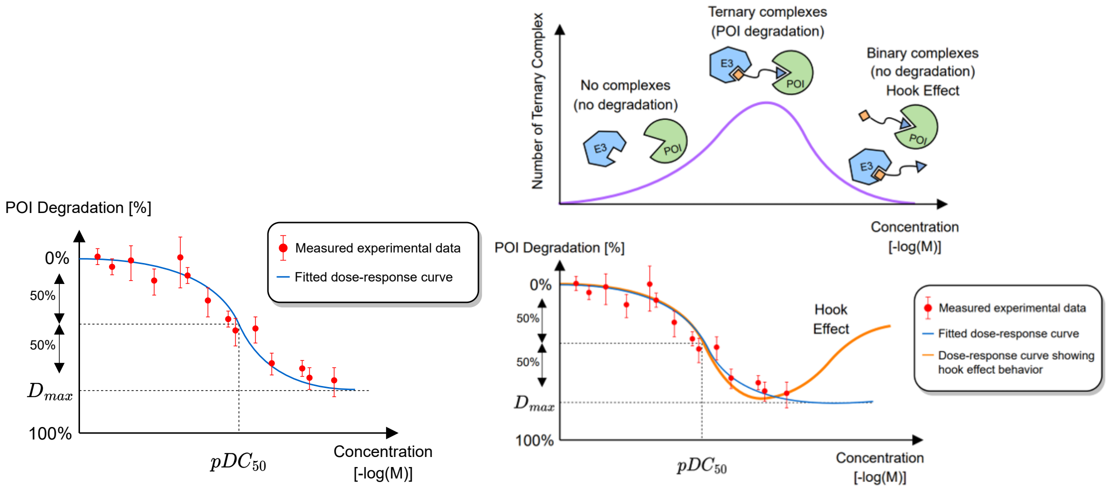
Figure 2.1: Visual representation of the ubitiquination and proteolysis processes degrading: a defective protein within a cell (top part of the figure), and a target POI in a PROTAC system (bottom part of the figure). Ubiquitin proteins are referred as “Ub”.

Despite straightforward in theory, in practice bringing the E3 ligase, the PROTAC, and the POI into close proximity to form ternary complexes is a challenging task [7]. In fact, due to the high number of actors and parameters involved into the degradation process, it is hard to precisely estimate the efficacy of a PROTAC molecule. Generally, one would desire to predict with high confidence the activity of a compound, in order to identify which ones most likely to be effective degraders and eventually removing the majority of candidates from needing to be synthesized and tested experimentally.

2.1.1 Measuring PROTAC Performance

One possible metric to assess PROTAC performance consists of the half-maximal Degradation Concentration 50% (DC_{50}, or pDC_{50} when expressed in negative logarithmic units, or Log_{10}(M)). The DC_{50} value measures the potency of a compound in degrading a target protein, where a lower value indicates that a compound is a more effective degrader. In particular, DC_{50} represents the molecular concentration at the inflection point of the dose-response curve. More precisely, DC_{50} is half of the concentration at the maximum degradation (D_{max}) achieved, as illustrated in Figure 2.2a.

The dose-response curve of a drug is typically described as a Hill function [8], as in



(a) Dose-Response curve.

(b) Dose-Response curve with hook effect.

Figure 2.2: Example of degradation activity versus concentration in a dose-response curve plot. (a) Dose-response curve fitted on experimental points. (b) Dose-response curve showing hook effect.

Equation 2.1, which has been fitted on data collected via laboratory assays[2].

$$f(x) = E_0 + \frac{EC_{inf} - E_0}{1 + exp\Big(\alpha\left(log(x) - log(IC_{50})\right)\Big)} \tag{2.1}$$

In the equation, E_0 is the observed response in absence of the drug, whereas EC_{inf} represents the maximum response caused by the drug. The IC_{50} value stands for the concentration required to inhibit 50% of a cell growth. On the other hand, the parameter α is associated to the slope of the function. By plugging in several measurements into the function (by substituting the x value), it is possible to interpolate the dose-response curve and estimate the IC_{50} and α values.

For PROTACs, the EC_{inf} value can be substituted by the D_{max} , whereas DC_{50} can mapped in the equation to IC_{50}. However, the equation is typically used to model the response of a given drug, and it's not tailored to PROTAC molecules. Because of that, the Hill equation modeling the dose-response curve for PROTACs presents some limitations, in particular when it comes to describe a phenomenon called *hook effect* [9], as shown in Figure 2.2b. The hook effect can happen if the formation of new ternary complexes decreases when the concentration of a given PROTAC increases. This is because the PROTAC, present in high concentrations, binds to either the available E3 ligase or POIs, forming binary complexes (PROTAC-POI and PROTAC-E3 ligase) preventing the formation of ternary complexes.

[2] Equation 2.1 reports the Hill function in a more numerically stable form which involves logarithms and exponents.

2.2 Embedding Vectors and Molecular Representations

In machine learning, data features and characteristics are often transformed into a continuous high-dimensional numerical representation called an *embedding vector*, or simply *embedding*. Embedding vectors can be seen as functions mapping information from a data point into a single n-dimensional real-value vector. One key advantage of dealing with embedding vectors rather than raw numerical data is that distances between different data points, and in turn their similarity, can be easily measured. Typically, embedding vectors are automatically generated via training ML algorithms. However, generating efficient embedding vectors able to model the relationships across data points can be challenging and generally depends on the selected data representation. Hence, it is fundamentally important to find a proper data representation that captures all relevant information one wants to feed into the ML algorithm. Because of that, we would need to find an efficient numerical representations of chemical molecules. In computational chemistry and bioinformatics, there exist several ways to represent molecules to be processed by computer systems. In the remaining part of this section, we describe the three common molecular representations which are most relevant for this research work.

2.2.1 Molecular Fingerprints

A simple yet effective representation of PROTAC molecules consists of *molecular fingerprints*. Briefly, a molecular fingerprint is a bit vector which describes, via on or off bits, specific molecule characteristics. Their key advantage compared to a bare chemical formula is that fingerprints allow for a more informative representation of the chemical structure of a molecule, making them suitable for computational algorithms and machine learning. There exist several algorithms for designing fingerprints, most notably: circular-based like Morgan fingerprints [10], based on structural keys like Molecular ACCess System (MACCS) [11], and path-based fingerprints [12]. A visual intuition of the aforementioned fingerprints types is shown in Figure 2.3.

One of the main limitations of molecular fingerprints is a possible loss of information when a bit-vector is not sufficiently long or when the molecule to encode presents many symmetries. In fact, in order to obtain the final bit-vector, fingerprint algorithms iteratively update a common set of characteristics of the atoms and their neighbours. Once the set is complete, a final hashing function applied to the set allows to extract the requested bit-vector. Molecular characteristics, however, might not be uniquely encoded in the process, for instance if the bit length assigned to the hash function doesn't guarantee uniqueness. Ultimately, this might lead to overlapping encodings for different characteristics, resulting in bit vectors for which one set bit can encode *multiple* characteristics (with the extreme case of having same fingerprints for different molecules).

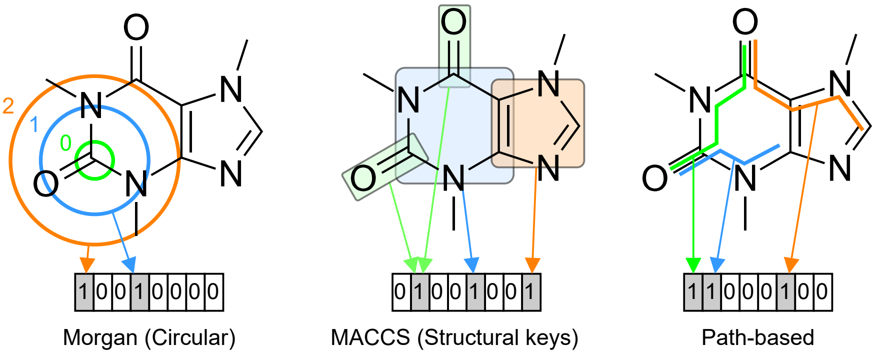


Figure 2.3: Examples of different fingerprint types. Inspired from [13].

2.2.2 Molecular Graphs

Graph theory can be applied to represent molecules and their structure. A graph consists of a pair of sets, the set of vertices, or nodes, V and the set of edges E. In the context of chemical graphs, each edge can be defined as a pair (a_i, a_j), usually representing a chemical bond between two atoms a_i and a_j in V. We limit our analysis to unidirected graphs, *i.e.*, where any edge (a_i, a_j) is equal to (a_j, a_i) for $i \neq j$, effectively expressing one chemical bond per atom pair, no matter their order. We also assume graphs with no self-directed edges, *e.g.*, with no edges (a_i, a_j) when $i = j$.

Given the above statements, graphs, and in turn molecules, can then be described by special matrices characterizing relationships between atoms. One example is the $n \times n$ *adjacency matrix*, whose entries (i, j) are zeros except from when there exists an edge connecting the atom pair (a_i, a_j), for any of the n atoms in V. Another useful graph encoding is the *degree matrix*, which includes the degree of each atom, defined as the number of edges connected to it.

Finally, each atom and/or bond can be associated to a set of features, such as atom type or bond type, atom weight, *et cetera*.

2.2.3 SMILES

Chemical graphs and their corresponding matrices are a comprehensive and efficient way of representing molecules. However, a more compact way is given by Simplified Molecular-Input Line-Entry System (SMILES) [14], a character string that describes their structures. Many valid SMILES strings can be generated from a given molecule. On the other hand, each SMILES always uniquely encodes a specific molecule. A classic example is the ethanol molecule, C_2H_6O, which can be written as `CCO`, `OCC`, and `C(O)C`.

First, cycles are broken down in order to create a spanning tree of the molecule, *i.e.*, any acyclic graph that includes all atoms. At this point, the tree is depth-first traversed and symbol nodes, like atoms and bond types, are sequentially added to the SMILES string. Additionally, extremes of cycles are numerically labelled, while

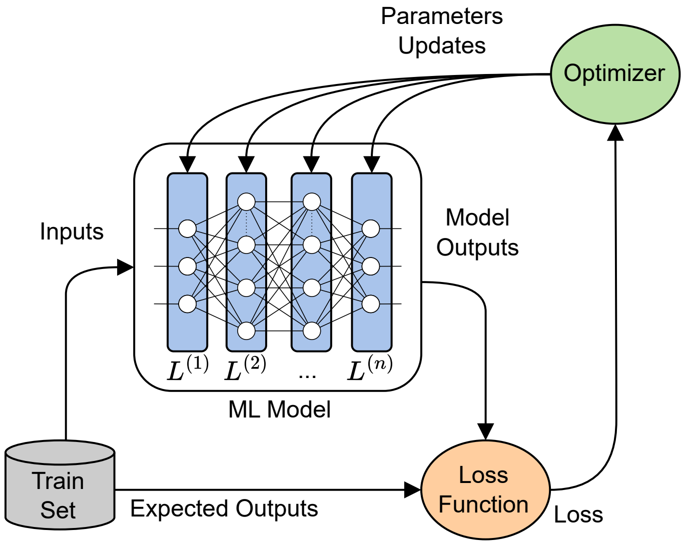


Figure 2.4: Typical flow for training a ML model given a training set of data.

branches are enclosed by nested parentheses.

Compared to a graph representation, SMILES strings result in an efficient and compact way of encoding molecular structures, allowing the use of Natural Language Processing (NLP) techniques.

2.3 Deep Learning

Machine learning and deep learning algorithms fall under the umbrella of Artificial Intelligence (AI). Deep Learning (DL) in particular comprises statistical models usually consisting of several interconnected layers, which process the incoming flow of data to approximate a desired function. To do so, each layer applies to its incoming input a differentiable function, which typically includes a set of trainable, *i.e.*, tunable, parameters. The obtained output values are then fed as inputs to the next layer, until reaching a final model output.

The DL model performance depends on a careful selection and tuning of its parameters. In order to tune the model parameters, and to *learn* a specific task to perform, the model needs to be *trained*, and in a supervised setting it requires to process a large amount of training data. In summary, during training the input-output data pairs are used to minimize a *loss function*, which depends on the expected desired output and the generated output from the model, which in turns depends on its internal parameters. By adjusting the model parameters, for example via stochastic gradient descent or any suitable optimizer algorithm, the loss function can be minimized, allowing the model to effectively map given inputs to the desired outputs encountered during training. The overall process is illustrated in Figure 2.4.

The structure and layer types of a model typically vary according to the end task the model is trying to achieve. In our specific setting, *i.e.*, AI applied to drug design, a sequence of layers tailored to capture molecular features is usually employed within a model, like graph-based layers.

In addition to more sophisticated layers, reported hereafter, the simplest layer type that can be thought of is a linear operation, followed by a non-linear function, referred as activation function, for example a Rectified Linear Unit (ReLU) operation [15]. Without losing generality, and avoiding some mathematical details for brevity, Equations 2.2 and 2.3 reports a Linear and a ReLU layers, respectively, when processing a vector input $x \in \mathbb{R}^n$. Connecting n of these layers pairs together back-to-back form a Multi-Layer Perceptron (MLP) architecture, which function is highlighted in Equation 2.4.

$$Linear(x) = b + \sum_{i,j} x_i w_{ij} \tag{2.2}$$

$$ReLU(x) = \max\{0, x\} \tag{2.3}$$

$$MLP(x) = ReLU^{(n)}(Linear^{(n)}(...ReLU^{(1)}(Linear^{(1)}(x))...)) \tag{2.4}$$

Despite being simple, MLP models are fairly powerful and versatile and used in many practical applications [16]. However, they might not efficiently capture latent input features, like relationships between elements of the same input, *e.g.*, between any x_i and x_j, $i, j \in \{1, ..., n\}$. More specifically, if we would like to predict specific characteristics of a molecule, their 2D and 3D structure, such as atom bonds for instance, would be better encoded via graph-based layers, like Graph Convolution (GC) or Graph ATtention (GAT) layers.

2.3.1 Graph Neural Networks

DL models processing graph data are typically referred as Graph Neural Networks (GNN). GNNs leverage graph representations, such as adjacency matrix and node or edge features, to make predictions about or label each node or the graph as a whole. They do so by iteratively updating node embeddings by aggregating information coming from neighboring nodes. The aggregation is typically referred as *message passing* and it is performed by specialized layers. One example are GC layers [17], which compute a weighted sum of the features of the neighboring nodes, before passing it through a non-linear activation function, similarly to what is done by a Linear layer followed by a ReLU one.

Mathematically, an adjacency matrix A of a graph is a square matrix indicating whether two nodes are connected together by an edge. The adjacency matrix has zeroes on the main diagonal, unless nodes have self-connections. Hence, the identity matrix I is added to the adjacency matrix in order to construct a new square matrix $\hat{A}$ which includes each node contributions. Given $\hat{A}$ and a diagonal matrix $\hat{D}_{ii} = \sum_j \hat{A}_{ij}$, we can define the output H of the l-th GC layer as in Equation 2.5.

$$H^{(l)} = \sigma\Big(\hat{D}^{-\frac{1}{2}}\hat{A}\hat{D}^{-\frac{1}{2}}\, H^{(l-1)}W^{(l)}\Big) \tag{2.5}$$

In Equation 2.5, σ stands for a generic non-linear function and $W^{(l)}$ for a set of learnable parameters. Computing $\hat{D}^{-\frac{1}{2}}\hat{A}\hat{D}^{-\frac{1}{2}}$ is a way of scaling and normalizing the adjacency matrix in order to prevent exploding/vanishing gradients when used in a deep neural network model, as described in [17].

This message passing operation is then repeated by stacking together multiple of such layers. In this way, the embedding of one node is progressively updated by the information coming from more and more distant nodes. A variant of message passing employs *attention* and *self-attention mechanisms* to better capture graph and nodal information. Attention can be summarized as a way of learning the different and most relevant relationships among the sub-elements of a data point while filtering out noise or irrelevant information. For instance, in NLP, words positioned at opposite ends of a sentence might have long-term dependencies, *e.g.*, can be linked together by a subject-verb-object relationship. More specifically, in GAT layers [18], the neighboring nodes contributions z_i of a node i are aggregated according to Equation 2.6:

$$z_i = \sum_j \alpha_{ij} h_j \tag{2.6}$$

where h_j is the feature vector for neighboring node j, and α_{ij} is an attention weight between node i and j. Attention weights can be computed using a Softmax function [19], as specified in Equation 2.7:

$$\alpha_{ij} = \frac{Softmax(e_{ij})}{\sum_k Softmax(e_{ik})} \tag{2.7}$$

where e_{ij} is a scalar value that measures the similarity between the feature vectors of nodes i and j, as defined in Equation 2.8

$$e_{ij} = LeakyReLU(a^T[Wh_i||Wh_j]) \tag{2.8}$$

where a, W are learnable parameters, || denotes concatenation, and LeakyReLU is a variant of the ReLU activation function [20].

Attention mechanisms, and self-attention in particular, are also extensively exploited by *Transformer* models. First proposed in 2017 [21], *Transformers* models and their building blocks have gained widespread popularity for their ability to efficiently encode cross-information within their inputs.

2.3.2 Transformers

SMILES can be viewed as a series of symbols describing parts of a molecular graph, similarly to how groups of words form a sentence. Because of that, language models can be leveraged to process SMILES strings. In order to do so, the first step is to utilize a *tokenizer* to break down, *tokenize*, a given SMILES string into *tokens*, which can then be fed as input to a Transformer model, which is particularly suited for processing sequence of data. Tokenization can be learned by ML models via statistically identifying patters in the training data, *i.e.* the SMILES strings one desires to process. Once an input SMILES is tokenized, it can be processed in parallel via self-attention mechanisms, effectively extracting the most important information out of it. The architecture of Transformer models typically includes an *encoder* and a *decoder*. The encoder, in particular, is in charge of transforming the input tokens into a sequence of *hidden states*, *i.e.*, embeddings.

One of the most popular Transformers encoders is BERT (Bidirectional Encoder Representations from Transformers)[22], which was originally pre-trained on a large corpus of text data. What sets BERT apart from previous language models is that it is bidirectional, meaning it takes into account both the left and right context of a sentence when generating embeddings. This allows it to capture more complex relationships between tokens, effectively improving its performance. BERT and in general pre-trained Transformers are particularly suited to be *fine-tuned* for tasks different from the ones they have been trained for. By fine-tuning, a pre-trained model can be further trained for a specific task, usually requiring less extensive training and smaller datasets.

In conclusion, linear, graph-based, and attention-based layers constitute the main building blocks employed in this project, giving us the foundations for building the more advanced models proposed in the later chapters of the thesis.

3

Related Work

This chapter gives a brief overview of the recent and current research works most relevant to ours.

As PROTACs are relatively novel, not many publications cover the use of machine learning techniques applied to PROTACs. On the other hand, most of the related work focuses instead on the prediction of the half maximal Inhibitory Concentration (IC_{50}) score of ligands, and its dual pIC_{50} expressed in negative logarithmic units, which measures how well molecules bind to a target protein. Given this formulation, IC_{50} can also give a quantitative estimation of the potency of a candidate molecule at inhibiting target proteins. Ligands with typically high pIC_{50} are defined as *active*.

In this regard, measuring the IC_{50} value can also be applicable to PROTACs. In fact, PROTACs need to bind to both an E3 ligase and a POI, therefore each binding can be measured by its own IC_{50} value. In turn, knowing the different IC_{50} values of a given PROTAC might still be useful for predicting its degradation activity.

In recent years, multiple GNN architectures have been developed for predicting binary activity and efficacy of the complexes formed by a ligand and a protein's structure. Knutson *et al.* [23] propose two GNN models to assess the activity and pIC_{50} score of protein-ligand complexes. The first model GNN_F is designed to encode the *features* from the protein docking of the ligands, whereas a second one, GNN_P, takes as input the protein and the ligand separately. The proposed architectures are able to accurately predict pIC_{50} and protein-ligand activity, but also show some limitations. In fact, despite providing lower RMSE compared to GNN_P, GNN_F requires expensive preprocessing via protein-docking calculations. On the other hand, GNN_P is simpler, but does not achieve the same level of RMSE. On top of that, their solution only focuses on predicting pIC_{50} , which is not ideal for assessing the biological activity of PROTAC molecules.

The work of Moesser *et al.* [24] proposes instead to encode protein-ligand complex information via different representations, namely: fingerprints or molecular graphs, and protein sequences. This information, after feature extraction, is then processed by different deep networks in separate "branches", before being concatenated together to produce a prediction as output. However, their work is not specifically designed for PROTACs and PROTACs complexes.

Closest to our work is DeepPROTACs, from Li *et al.* [7], a deep learning model for predicting PROTAC activity. In DeepPROTACs, PROTAC molecules are defined as active when their D_{max} reaches more than 80% and their DC_{50} is below 100 nM (equivalently to pDC_{50} greater than 7). The model consists of several LSTM- and GNN-based branches concatenated together before a prediction head. Each branch processes different features of the PROTAC-POI complex, for instance the E3 ligase and POI binding pockets, and the separate PROTAC components, *i.e.*, warhead, linker and E3 ligand. The model achieves 77.95% average prediction accuracy and 0.8470 ROC-AUC score on a test set extracted from the PROTAC-DB.

Despite its performance, the model presents several limitations. First, it is possible that some information is lost when the PROTAC SMILES is split in its E3 ligand, warhead and linker and fed into three different model branches. Second, the authors perform advanced molecular docking of the whole PROTAC-POI-E3 ligase complex, but then only consider the information of the 3D binding pockets of the POI and E3 ligase, making it difficult to experimentally reproduce and use. Third, the model was not compared to other state-of-the-art and high performance models, such as XGBoost. Finally, the authors were not careful to prevent a possible data leakage in the hyperparameter search and training process, eventually requiring more accurate testing. For all these reasons, we believe that the DeepPROTACs model might not be suitable for accurately predict target protein degradation and for developing other downstream tasks, like PROTAC molecular generation [4].

4

Methods

In this chapter we report our research methodology. We start by detailing the data curation process we performed. We then proceed to list our candidate models for predicting targeted protein degradation. Finally, the chapter concludes with a description of the experimental setup.

4.1 Data Curation Process

Our work is based on public data coming from two regularly updated datasets: PROTAC-DB [5], which contains around four thousands PROTAC complexes and data web-scraped from related scientific literature, and PROTAC-Pedia [6], including around one thousands crowd-sourced entries about PROTAC molecules, their characteristics and degradation activity. In order for a PROTAC complex to be processed by our models, we extracted numerical features from relevant information such as: molecular SMILES, the cell type in which the PROTAC was tested, E3 ligase, POI sequence, and its degradation performance in terms of pDC_{50} and D_{max} .

Each entry of the datasets reports the SMILES string of the PROTAC molecule. All SMILES are converted to canonical representation and their stereochemistry information is removed. In PROTAC-DB, most of the entries present cell type information in the textual assay description, for example: "degradation in LNCaP cells after 6 h at 0.1/1000/10000 nM". Such information are parsed and extracted via a Named Entity Recognition (NER) system based on a BERT Transformer [25] (the treatment time is also extracted but not considered in our work). We encoded the various cell types via a SKlearn ordinal encoder, so that each type is mapped to an integer class. Another SKlearn ordinal encoder[1] was used to encode the PROTAC complex E3 ligase in a similar fashion. The POI amino acid sequence for each entry were instead web scraped. We then applied gene mutations by substituting and/or removing amino acids (represented as characters) or sub-sequences when reported in the entry. Finally, we employed a SKlearn count vectorizer to store the counts of m-to-n-grams of the mutated POI sequence into an integer vector.[2]

Regarding PROTAC degradation activity, we follow the work of DeepPROTACs[7]

[1] https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.OrdinalEncoder.html

[2] For instance, if considering one- and bi-grams, a count vectorizer would return a concatenated vector of the counts of each amino acid, together with the count of each possible combination of pairs of amino acids that can occur in the POI sequence.

in defining entries with pDC_{50} higher than 7 and D_{max} higher than 80% as *active*, *inactive* otherwise. All PROTAC-Pedia entries report being either active or inactive. However, their definition of "being active" is less conservative, being pDC_{50} higher than 6 and D_{max} higher than 30%. Because of that, we calculated the DeepPROTACs active/inactive definition for those entries reporting DC_{50} and D_{max} information. On the other hand, PROTAC-DB required additional effort in extracting pDC_{50} and D_{max} values for determining PROTAC activity. In fact, PROTAC-DB entries reported degradation activity in either one of two pairs of columns:

- either concentrations and degradation percentages that describe points on the dose-response curve, combined with the assay description,
- or DC_{50} concentration and D_{max} percent degradation, together with the corresponding assay description.

For the first type of information, the DC_{50} value is not directly reported, so it would be required to estimate it via a dose-response curve fitting of the data, provided there are enough data points in the entry. For lack of time, we are only considering entries which report the second type of information, *i.e.*, the DC_{50} & D_{max} pair, as performing dose-response curve fitting for PROTAC complexes is challenging and not straightforward due to the hook effect, as mentioned in Section 2.1.

In the end, once the data curation process is complete, each data point consists of a binary variable for it being active/inactive, a PROTAC SMILES string, the respective class identifiers of cell type and E3 ligase, and an integer vector for the POI sequence m-to-n-grams counts.

4.2 Proposed Deep Learning Model

One of the main challenges of the thesis work is to efficiently encode information about PROTAC-POI-E3 ligase complex to be processed by a DL model. Following previous work on predicting Protein-ligand interaction and DeepPROTACs [7], we propose a model architecture which concatenates the embeddings produced by different model branches, each one processing different features of the complex. The respective embedding vectors are eventually concatenated before being fed into a final MLP or Linear head. An overview of the model architecture is depicted in Figure 4.1.

The E3 ligase and cell type integer classes are concatenated together with the POI m-to-n-grams as an input to an MLP model, as shown in the bottom right part of Figure 4.1. On the other hand, PROTAC molecules have several possible representations, as described in Section 2.2. Hence, we propose different model architectures for generating PROTAC embeddings, each one tailored to a specific molecular representation. Since all the considered molecular representations are derived from SMILES, we refer to this model branch as SMILES encoder. Different SMILES encoder architectures are summarized in Figure 4.2.

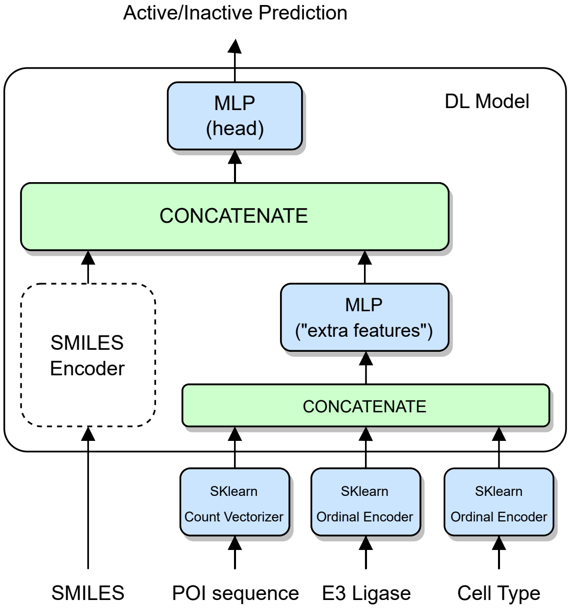


Figure 4.1: Proposed deep learning model architecture. The PROTAC SMILES string is processed by a specialized SMILES encoder. On the other hand, an MLP model is in charge of processing the numerically encoded extra features of the complex, such as the cell type, POI sequence and E3 ligase classes. The outputs of the two model branches are concatenated together before being fed to an additional MLP model, which is in charge of computing the final predictions.

4.2.1 PROTAC Encoding via Molecular Fingerprints

Starting from the PROTACs SMILES, we generate Morgan, MACCS and path-based fingerprints by using the RDKit library toolkit [26]. Since molecular fingerprints are bit vectors, we propose a simple MLP model for extracting PROTAC embeddings. The sub-model is depicted in Figure 4.2a.

4.2.2 PROTAC Encoding via Molecular Graphs

Since PROTAC molecules can be represented as chemical graphs, we employed the Pytorch Geometric library [27] to generate their respective graph encodings. In particular, the SMILES of each data entry is encoded as an adjacency matrix and node features matrix. A GNN-based sub-model is then employed to process graph encodings, as showed in Figure 4.2b.

4.2.3 PROTAC Encoding via SMILES String

PROTACs SMILES strings can be processed by Transformer-based models, such as BERT. For this proposed model instance we rely on pre-trained BERT-based Transformers, such as [28]. The selected BERT model can then be fine-tuned on our specific task, eventually learning a suitable PROTAC embedding representation, which can be extracted from the CLS output head of BERT[22]. The Transformer-

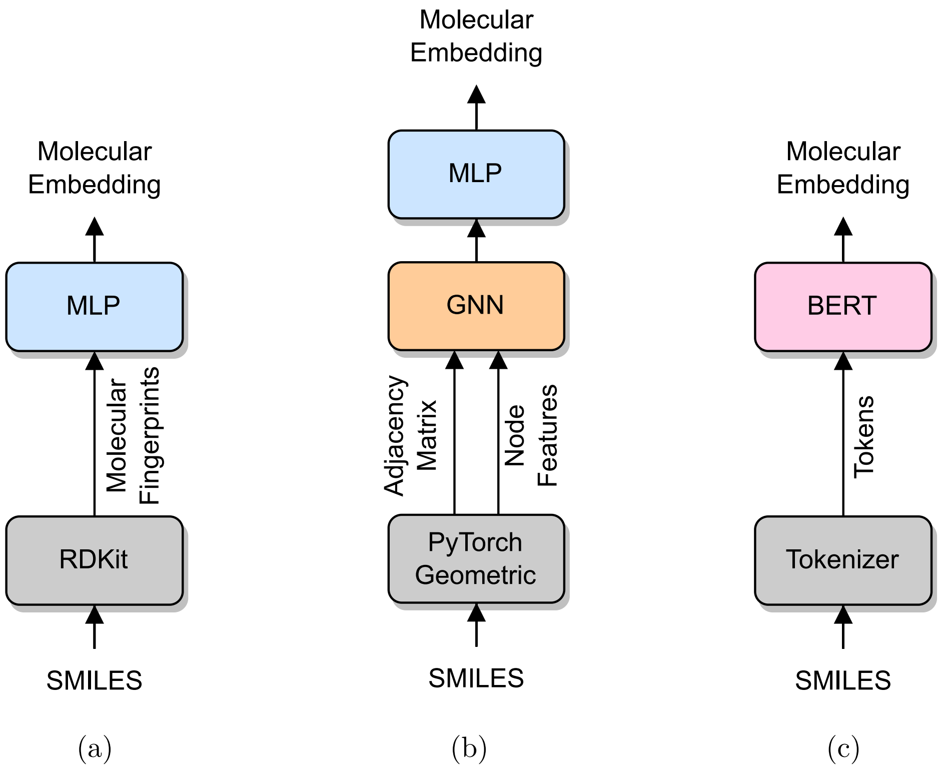


Figure 4.2: Proposed architectures for the SMILES encoder module. The SMILES transformations and feature extractions depicted in the gray modules are all done offline outside the DL model. (a) Molecular fingerprint encoder. (b) Molecular graph encoder. (c) BERT-based encoder.

based SMILES encoder can be viewed in Figure 4.2c .

4.2.3.1 Self-Supervised Learning and Transfer Learning

Self-Supervised Learning (SSL) is a machine learning technique that allows models to learn from unlabeled data [29]. As explained in Section 2.3, in traditional supervised learning, models are trained on labeled datasets where each input is associated with a corresponding target or label. However, in SSL, a given model is trained to predict certain properties or relationships within the data itself, without the need for explicit labels. In doing so, the model learns the inner relationships between the data, and in turn their latent features.

SSL offers a potential solution to the lack of large training datasets by leveraging the abundance of unlabeled data available. As better highlighted later in Chapter 5, Table 5.1, most of the entries in PROTAC-DB and PROTAC-Pedia are in fact missing the required information for marking them as active or inactive. Because of that, we decided to experiment with SSL applied to Transformer-based models. This is typically done by creating auxiliary tasks that require the model to predict missing parts of the input, such as masked words in a sentence or missing pixels in an image. The model is then trained to reconstruct the original input, effectively learning to capture high-level features and structures within the data.

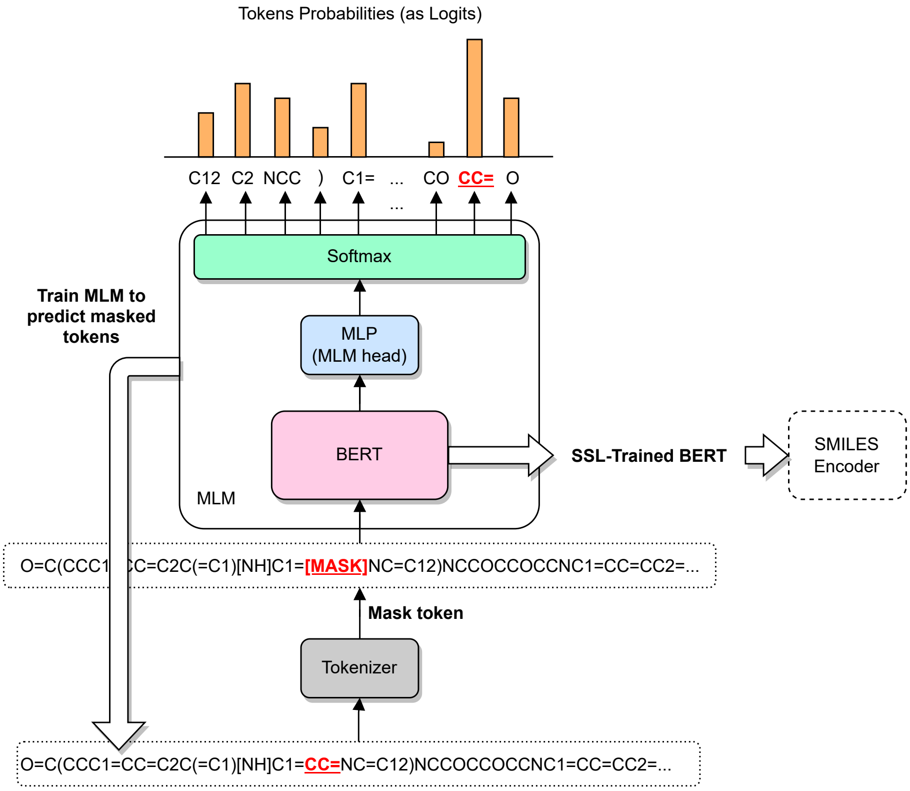

Tokens Probabilities (as Logits)
C12
C2
NCC
)
C1=
...
CO
CC=
O
...
Softmax
MLP
(MLM head)
BERT
MLM
Train MLM to predict masked tokens
SSL-Trained BERT
SMILES Encoder
O=C(CCC1=CC=C2C(=C1)[NH]C1=[MASK]NC=C12)NCCOCCOCCNC1=CC=CC2=...
Mask token
Tokenizer
O=C(CCC1=CC=C2C(=C1)[NH]C1=CC=NC=C12)NCCOCCOCCNC1=CC=CC2=...

Figure 4.3: SSL training of a BERT-based model wrapped in a MLM model. The MLM is trained to predict the masked tokens by outputting a log-likelihood over all possible tokens. Once SLL is complete, the BERT model can be extracted and finetuned on other downstream tasks, *e.g.*, as a SMILES encoder.

A detailed overview of SSL training applied to a Transformer BERT-based model is illustrated in Figure 4.3. The idea is to wrap a BERT model into a Masked Language Model (MLM), effectively taking the embeddings generated by BERT and forwarding them to a MLP head first and finally to a Softmax function. In the example of the figure, the token `CC=` is masked out and the model is trained to predict it back by assigning to the masked token the highest probability (expressed as logits) among all possible tokens.

Once the model is pre-trained using SSL, it can then be extracted from MLM model as is and fine-tuned on specific downstream tasks using a smaller labeled datasets. In our case, the SSL model is integrated as a SMILES encoder in our proposed model. This transfer learning approach allows the SMILES encoder to leverage the learned representations from the self-supervised phase, ideally improving the performance of the overall model even with limited labeled data.

In our work, we assembled an SSL dataset from the discarded entries of PROTAC-DB and the ones assigned to the training set, leaving out the data from the validation dataset and from PROTAC-Pedia, which is used for assembling the test set. Then, we proceeded to swap and mask out around 15% of tokens obtained from each molecular SMILES of the SSL dataset data. Finally, we trained the candidate Transformer

Table 4.1: The baseline model and the candidate SMILES encoder models.

Baseline and SMILES Encoders	Design Point
XGBoost baseline	Fingerprints at 1024 bit Fingerprints at 2048 bit Fingerprints at 4096 bit Fingerprints at 1024 bit w/ extra features Fingerprints at 2048 bit w/ extra features Fingerprints at 4096 bit w/ extra features
MLP-based	Fingerprints at 1024 bit Fingerprints at 2048 bit Fingerprints at 4096 bit
GNN-based	AttentiveFP [33] GAT [18] GCN [17] GIN [34]
Transformer-based	Roberta_zinc_480m [35] ChemBERTa-zinc-base-v1 [36] ChemBERTa-10M-MTR [37] SSL_Roberta_zinc_480m [35] SSL_ChemBERTa-zinc-base-v1 [36] SSL_ChemBERTa-10M-MTR [37]

models to predict the original SMILES string.

4.3 Experimental Setup

In order to evaluate the proposed model, consisting of different SMILES encoder architectures, we developed a PyTorch Lightning module following the design of Figure 4.1. Within the module, another PyTorch module representing the SMILES encoder is instantiated. [3]

Both the model and its SMILES encoder modules present a series of hyperparameters, like the number of layers and the their size, for example, that require a careful selection in order to obtain the best model performance. Because of that, we employed the Optuna optimization framework [30] for tuning the models' hyperparameters. For each candidate model, Optuna was setup to use an Hyperband optimization scheduler and a TPE sampler to find the best hyperparameters combination among 1,000 trials.

We used an XGBoost-based model [31] as a baseline to compare DL models to. It leverages a series of weak learners, *e.g.*, trees, to make predictions in a ensembled fashion. XGBoost is a popular and high performing model [32] based on gradient boosting and as such it was chosen as a control model in our evaluation.

All the candidate SMILES encoder together with variations of the baseline model are listed in Table 4.1. The design points reflect some hyperparameters that we

[3] The full implementation code can be viewed at: https://github.com/orgs/ailab-bio.

fixed, *i.e.*, did not let Optuna optimize. For example, in case of XGBoost- and MLP-based models, we did so to investigate and compare the effects on performance of certain molecular fingerprint bitwidths. In particular, XGBoost baselines were evaluated both including and not including extra features such as the E3 ligase and cell types and the POI m-to-n-grams counts. In case of GNNs instead, we wanted to compare different popular GNN architectures [17], [18], [33], [34]. Similarly, for the Transformer-based SMILES encoders, we selected three popular BERT models from the Hugging Face hub [38]: Roberta_zinc_480m [35] (57,920 downloads in May 2023), ChemBERTa-zinc-base-v1 [36] (10,139 downloads in May 2023) and ChemBERTa-10M-MTR [37] (974 downloads in May 2023). The Transformer models are also included after performing SSL training on each one of them (in the table, they are marked with the "SSL_" prefix).

A complete list of hyperparameters ranges, *i.e.*, the Optuna search space, can be viewed in Appendix A.

5

Results and Discussion

This chapter reports the results obtained by following the methodology described in Chapter 4. It also includes a comprehensive discussion of the gathered experimental results.

5.1 Data Distributions

As we can see in Table 5.1, the data curation process discarded most of the data points in PROTAC-DB and PROTAC-Pedia, removing 90.8% and 86.4% of the entries, respectively. Since the number of curated entries is higher for PROTAC-DB, we decided to use it as train and validation dataset, whereas PROTAC-Pedia is treated as a test set. We randomly split PROTAC-DB assigning 80% of its entries to the training set and the remaining 20% to the validation set. In order to prevent data leakage, we further made sure that the same SMILES entry would not show up in both train and validation sets. On top of that, we removed from the PROTAC-Pedia test set all SMILES entries already present in the training set. Figure 5.1 reports the final number of active and inactive entries, *i.e.*, the binary labels, in the train, validation and test sets. Because of the number of inactive entries in the train set being only 12.8% more than the active ones, we did not deem it necessary to apply data balancing techniques such as data up-/down-sampling or using class weights.

Before continuing with the feature distributions, Figure 5.2 illustrates the obtained distribution of D_{max} versus pDC_{50} values (which define the active/inactive binary labels). Figures 5.2a and 5.2b refer to the curated data from PROTAC-DB and PROTAC-Pedia, respectively. For both datasets, as the pDC_{50} value increases, the range for D_{max} values gets progressively narrowed, suggesting a general trend where potent PROTACs, *i.e.*, with high pDC_{50} value, tend to higher degradation efficiency, *i.e.*, D_{max} values.

Table 5.1: Number of curated data extracted from the PROTAC-DB and PROTAC-Pedia datasets.

Dataset	**# Entries before data curation**	**# Entries after data curation**	**Difference**	**Removed / Remaining**
PROTAC-DB	5,388	426	4,892	90.8% / 9.2 %
PROTAC-Pedia	1,203	164	1,039	86.4% / 13.6%

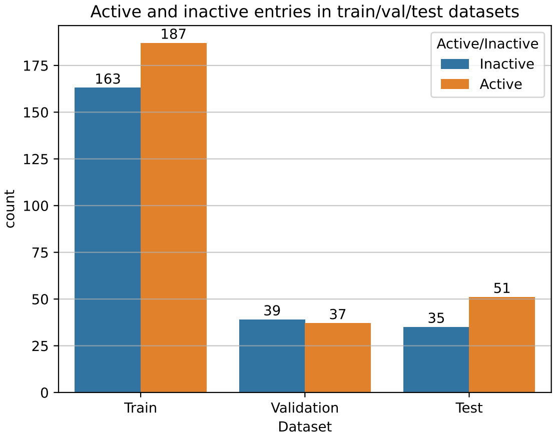
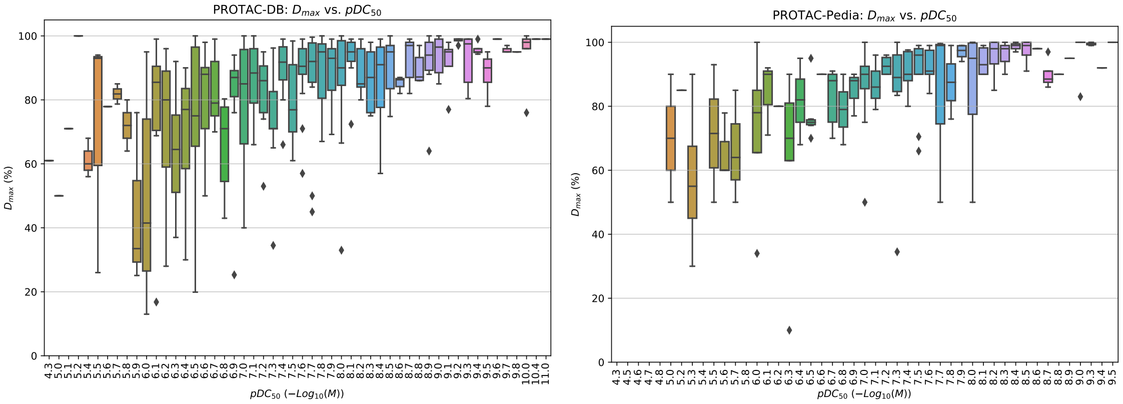


Figure 5.1: Number of active and inactive entries per train/val/test datasets.

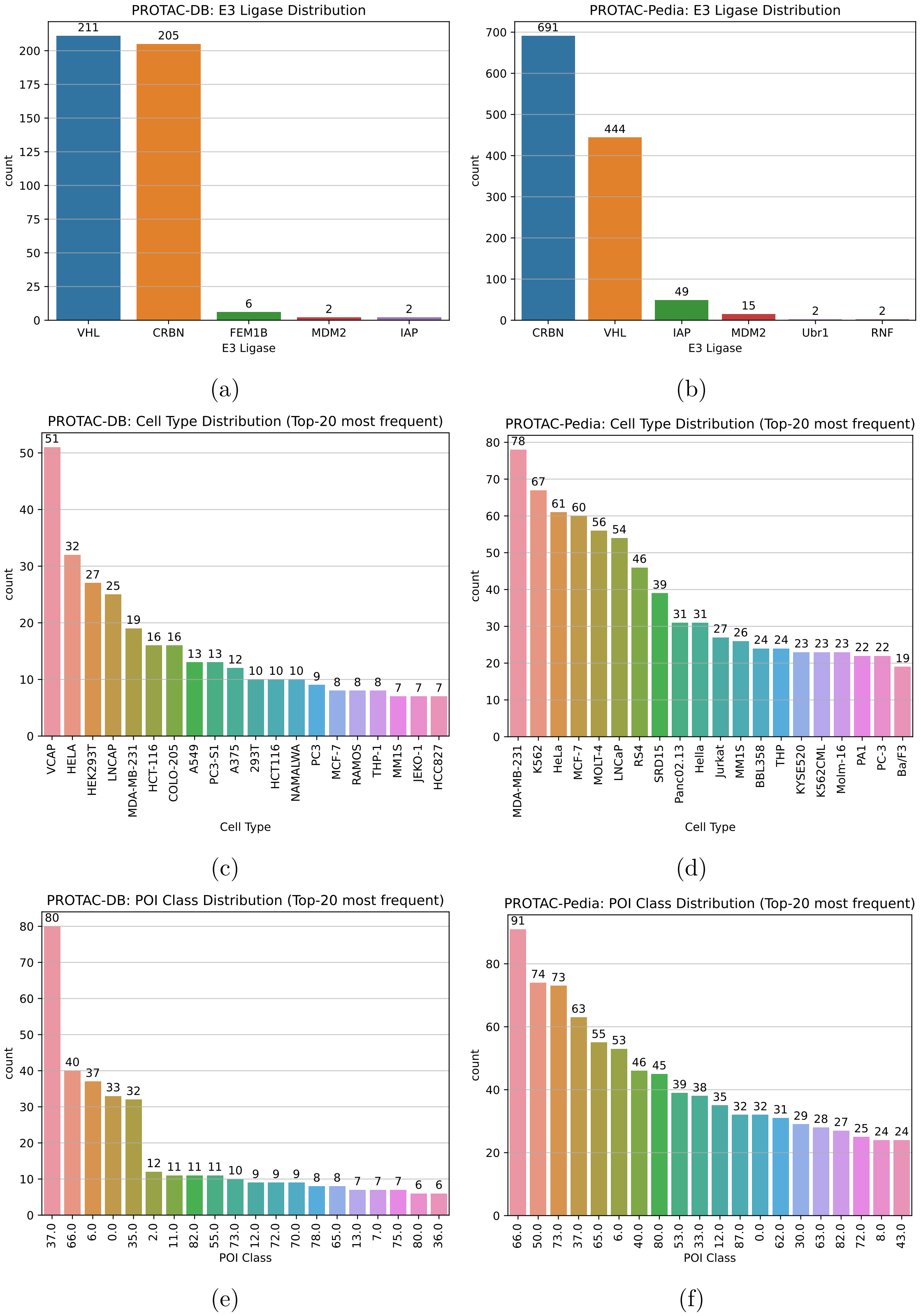


(a) PROTAC-DB.

(b) PROTAC-Pedia.

Figure 5.2: Distribution of D_{max} versus pDC_{50} values for PROTAC-DB and PROTAC-Pedia datasets after data curation.

Regarding the distribution of the model's input features, Figure 5.3 highlights the E3 ligase type, cell type and POI amino acid class distributions, respectively, for both the curated PROTAC-DB and PROTAC-Pedia. We are showing the distributions for the entire curated datasets, as they provide better insights compared to the same distributions withing the train/val/test sets. Since the train/val/test sets were randomly split, their resulting distributions are anyway similar to the ones of the entire curated datasets. There is a class imbalance for the VHL and Cereblon (CRBN) types in both PROTAC-DB and PROTAC-Pedia datasets and is due to the VHL and CRBN E3 ligase being the most common ones referred in the literature, as shown in Figures 5.3a and 5.3b. If we look at the most common cell types instead, in Figures 5.3c and 5.3d, apart from Hela, PC3 and LNCAP cells, the two datasets include different cell type in their top-20 most frequent ones. A similar observation can be made for the most common POI, shown in Figures 5.3e and 5.3f after their amino acid sequences have been encoded via a SKlearn ordinal encoder. Out of the top-20 most frequent POIs, ten are common between PROTAC-DB and PROTAC-Pedia.



(a) (b) (c) (d) (e) (f)

Figure 5.3: Distributions of the model's input features. (a) E3 ligase classes in PROTAC-DB. (b) E3 ligase classes in PROTAC-Pedia. (c) Cell types in PROTAC-DB. (d) Cell types in PROTAC-Pedia. (e) Top-20 most frequent POI classes in PROTAC-DB. (f) Top-20 most frequent POI classes in PROTAC-Pedia.

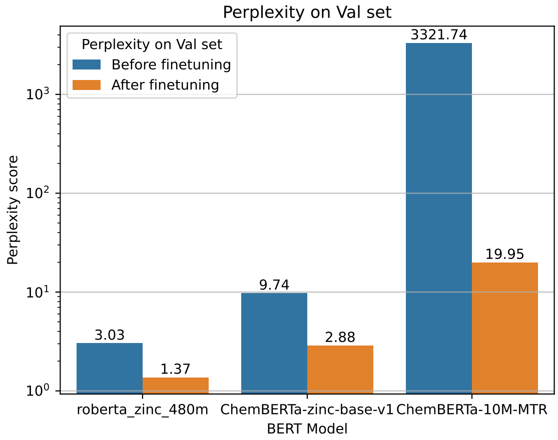


Figure 5.4: Perplexity score before and after SSL finetuning of the candidate Transformer models.

5.2 Self-Supervised Learning Perplexity Scores

Before showing the experimental results and comparison of the different candidate models we considered, this section includes the perplexity scores achieved by the Transformer models trained in a SSL fashioned. Perplexity is a measure of how well a language model predicts a given sequence of words or tokens. It quantifies the uncertainty or "perplexity" of the model when trying to predict the next word or token in a sequence. Perplexity can be estimated via Equation 5.1.

$$\text{Perplexity} = exp\Big(-\frac{1}{N}\sum_{N}\sum_{i} log\Big(P(t_i|t_1, t_2, ..., t_{i-1})\Big)\Big) \tag{5.1}$$

In our case, N represents the number of SMILES in the evaluation dataset, whereas t_i denotes the i-th token in the sequence. $P(t_i|t_1, t_2, ..., t_{i-1})$ is the probability assigned by the language model to predict the i-th token given the preceding tokens in the given sequence, *i.e.*, SMILES. Such probabilities are returned by the language model as logits, *i.e.*, $log\Big(P(t_i|t_1, t_2, ..., t_{i-1})\Big)$, and therefore they need to be summed for each evaluated SMILES (also refer to Figure 4.3 for a visual representation of the model output).

The perplexity formula essentially calculates the average negative log-likelihood of the tokens in the tokenized SMILES sequences according to the language model[1]. By exponentiating the result, one can obtain a perplexity value that is easier to interpret, with lower values implying that the model is more confident about its predictions, thus suggesting better predictive performance.

Figure 5.4 shows the perplexity score on the validation set before and after finetuning the three candidate Transformer models via SSL. We can observe that for all models

[1] Using logarithms in the computation also helps with numerical stability. For example, while multiplying together probabilities, having their logarithms turns the operation into a sum, thus preventing that one very small probability rounds the whole result to zero.

Table 5.2: XGBoost optimal model combinations of fixed hyperparameters.

Hyperparameters	**Values**					
	M$_1$	**M**$_2$	**M**$_3$	**M**$_4$	**M**$_5$	**M**$_6$
fp_bits (fixed)	167	1024	2048	2048	167	4096
extra_features (fixed)	True	False	True	False	True	False
fp_type	MACCS	Morgan	Morgan	Morgan	MACCS	Path
fp_radius	-	2	3	2	-	7
fp_max_path	-	-	-	-	-	10
booster	gbtree	gblinear	dart	gblinear	dart	dart
lambda	3e-07	0.005	9e-06	1e-06	0.003	4e-05
alpha	2e-08	5e-05	6e-08	2e-06	0.003	1e-06
max_depth	20	-	7	-	10	10
eta	0.03	-	0.0005	-	0.04	3e-06
gamma	0.004	-	1e-07	-	0.0005	7e-05
grow_policy	lossguide	-	depthwise	-	lossguide	lossguide
sample_type	-	-	uniform	-	weighted	uniform
normalize_type	-	-	forest	-	forest	tree
rate_drop	-	-	0.05	-	7e-05	1e-07
skip_drop	-	-	0.2	-	0.5	4e-06
Scores						
Accuracy [%] (Validation)	**81.58**	76.32	73.68	80.26	81.58	73.68
Accuracy [%] (Test)	56.98	51.16	53.49	48.84	56.98	**61.63**
ROC AUC (Validation)	0.8261	0.7980	**0.8611**	0.8136	0.8261	0.8278
ROC AUC (Test)	**0.7095**	0.5765	0.5692	0.5020	0.6983	0.6275

the perplexity score decreases after SSL training, with an improvement of two order of magnitudes for the ChemBERTa-10M-MTR model and of 2.21× and 3.38× for Roberta_zinc_480m and ChemBERTa-zinc-base-v1, respectively.

5.3 Experimental Results: Optimal Hyperparameters and Performance Scores

We here report the experimental results collected after following the experimental setup described in 4.3, *i.e.*, after hyperparameter optimization via Optuna and training of the candidate models.

Tables 5.2, 5.3, 5.4, and 5.5 show the found hyperparameters and performance scores for the XGBoost baseline, the MLP-, GNN-, and BERT-based models, respectively. In the tables, hyperparameters marked as “fixed” are not optimized during the optimization search and generate the combinations listed in Table 4.1. The different combinations of models with fixed hyperparameters are progressively marked as M$_1$, ..., M$_n$. Figures 5.5 and 5.6 show instead a comparison of the accuracy and Receiver Operating Characteristic Area Under the Curve (ROC AUC) scores, respectively, of the different models. A dummy score was added for both accuracy and AUC results. For accuracy, the dummy model always outputs the most frequent class in the dataset, meaning that its accuracy score will be equal to the percentage of most abundant class label. On the other hand, we assume perfect error for true

Table 5.3: Optimal hyperparameters of MLP-based SMILES encoder models found after Optuna optimization search.

Hyperparameters	**Values**		
	M$_1$	**M$_2$**	**M$_3$**
num_epochs (fixed)	10	10	10
num_layers_extra_features	5	3	4
layer_sizes	[480, 928, 224, 224, 448]	[352, 192, 288]	[480, 736, 672, 128]
dropout	0.6	0.2	0.3
learning_rate	0.002	0.0001	0.0003
batch_size	32	64	64
SMILES Encoder-specific			
fp_bits (fixed)	1024	2048	4096
fp_type	Morgan	Morgan	Morgan
fp_radius	2	3	5
fp_max_path	-	-	-
num_layers	2	2	2
layer_0_size	608	576	32
layer_1_size	768	352	704
dropout	0.2	0.6	0.2
Scores			
Accuracy [%] (Validation)	75.00	75.00	**80.26**
Accuracy [%] (Test)	55.81	**67.44**	54.65
ROC AUC (Validation)	**0.82**	0.80	0.81
ROC AUC (Test)	0.61	**0.68**	0.59

Table 5.4: Optimal hyperparameters of GNN-based SMILES encoder models found after Optuna optimization search.

Hyperparameters	**Values**			
	M$_1$	**M$_2$**	**M$_3$**	**M$_4$**
num_epochs (fixed)	50	50	50	50
num_layers_extra_features	4	2	3	2
layer_sizes	[288, 256, 192, 352]	[96, 64]	[384, 256, 96]	[448, 384]
dropout	0.12	0.20	0.10	0.07
learning_rate	8e-05	0.001	0.0003	2e-05
batch_size	8	8	8	4
SMILES Encoder-specific				
gnn_type (fixed)	Attentivefp	GAT	GCN	GIN
jk	-	last	last	lstm
hidden_channels	320	128	384	128
num_layers	8	6	3	3
dropout	0.59	0.39	0.66	0.61
out_channels	768	64	64	-
num_timesteps	64	-	-	-
accumulate_grad_batches	1	4	1	8
Scores				
Accuracy [%] (Validation)	69.74	68.42	**76.32**	72.37
Accuracy [%] (Test)	50.00	44.19	**62.79**	60.47
ROC AUC (Validation)	0.71	0.74	**0.74**	0.71
ROC AUC (Test)	0.50	0.46	**0.63**	0.63

Table 5.5: Optimal hyperparameters of BERT-based SMILES encoder models found after Optuna optimization search.

Hyperparameters	**Values**					
	$\mathbf{M}_1$	$\mathbf{M}_2$	$\mathbf{M}_3$	$\mathbf{M}_4$	$\mathbf{M}_5$	$\mathbf{M}_6$
num_epochs (fixed)	5	5	5	5	5	5
num_layers_extra_features	5	4	6	4	8	4
layer_0_size	384	512	224	384	288	480
layer_1_size	448	384	384	384	192	96
layer_2_size	384	320	384	288	512	192
layer_3_size	352	128	192	416	512	288
layer_4_size	64	-	192	-	512	-
layer_5_size	-	-	224	-	512	-
layer_6_size	-	-	-	-	128	-
layer_7_size	-	-	-	-	512	-
dropout	0.33	0.13	0.40	0.35	0.03	0.56
learning_rate	4e-05	1e-05	0.0002	3e-05	7e-05	0.0001
batch_size	4	4	8	4	8	8
SMILES Encoder-specific						
SSL (fixed)	False	False	False	True	True	True
bert_type (fixed) (reference)	[35]	[36]	[37]	[35]	[36]	[37]
accumulate_grad_batches	4	4	2	4	8	2
Scores						
Accuracy [%] (Validation)	72.37	**78.95**	76.32	76.32	77.63	73.68
Accuracy [%] (Test)	55.81	**63.95**	63.95	61.63	59.30	63.95
ROC AUC (Validation)	**0.87**	0.86	0.81	0.83	0.82	0.81
ROC AUC (Test)	0.57	0.62	0.61	**0.64**	0.61	0.59

and false positive outputs, leading to a 0.50 AUC score for the dummy model.

On average, XGBoost models achieved a validation accuracy of approximately 77.8% and a validation ROC AUC of 0.83. The best-performing XGBoost models achieved a validation accuracy of 81.58% and a validation ROC AUC of 0.83. On the test set, the average accuracy for XGBoost models was around 54.9%, with an average ROC AUC of 0.62. The best XGBoost model achieved a test accuracy of 61.6% and a test ROC AUC of 0.71.

The average accuracy on the validation set for MLP models was approximately 76.8%, with an average ROC AUC of 0.81. The best MLP models achieved a validation accuracy of 80.3% and a validation ROC AUC of 0.82. On the test set, MLP models achieved an average accuracy of 59.3% and an average ROC AUC of 0.62. The best MLP models achieved a test accuracy of 67.4% and a test ROC AUC of 0.68.

The average accuracy on the validation set for GNN models was around 70.8%, with an average ROC AUC of 0.73. The best GNN models achieved a validation accuracy of 76.3% and a validation ROC AUC of 0.74. On the test set, GNN models achieved

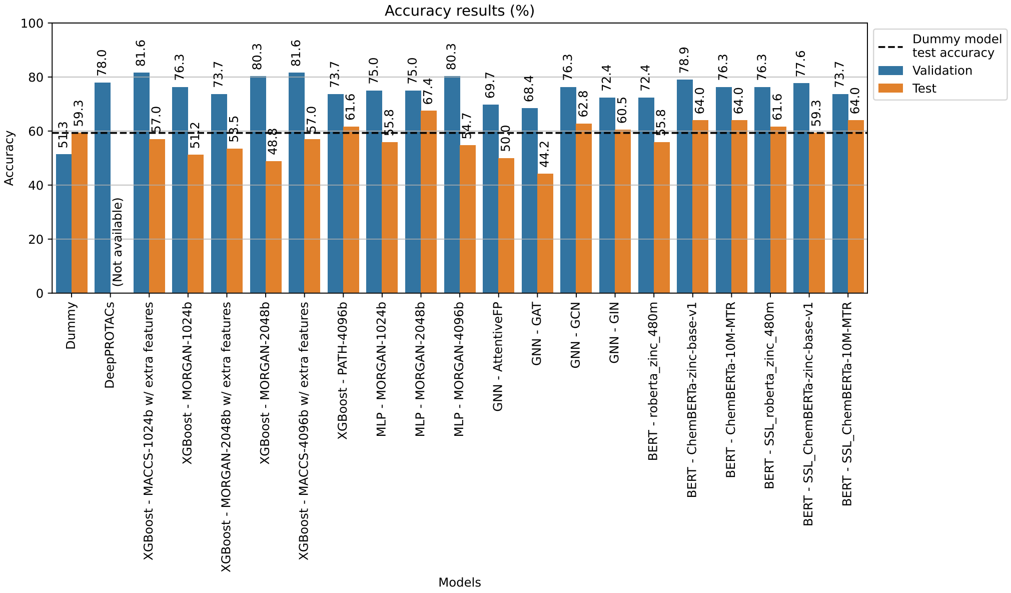


Figure 5.5: Accuracy scores of the candidate models. The higher the better.

an average accuracy of 51.9% and an average ROC AUC of 0.56. The best GNN models achieved a test accuracy of 62.8% and a test ROC AUC of 0.63.

Transformer models achieved an average accuracy of approximately 75.9% on the validation set, with an average ROC AUC of 0.83. The best Transformer models achieved a validation accuracy of 78.9% and a validation ROC AUC of 0.87. On the test set, Transformer models achieved an average accuracy of 58.67% and an average ROC AUC of 0.62. The best Transformer models achieved a test accuracy of 64.0% and a test ROC AUC of 0.64.

5.4 Discussion

The results indicate that XGBoost models achieve competitive performance across different feature representations, with the highest validation accuracy achieved by XGBoost - MACCS-1024b with extra features (validation accuracy of 81.58%) and the highest validation ROC AUC achieved by XGBoost - MORGAN-2048b (validation AUC of 0.861). MLP models also demonstrated promising performance, with MLP - MORGAN-2048b achieving a validation accuracy of 80.26%. GNN models, including AttentiveFP, GAT, GCN, and GIN, exhibited relatively lower performance compared to XGBoost and MLP models, indicating the potential limitations of graph-based approaches for the curated data. Similarly, Transformer models, such as BERT - Roberta_zinc_480m and BERT - ChemBERTa-zinc-base-v1, achieved moderate performance in terms of validation accuracy and ROC AUC.

5.4.1 Comparison with DeepPROTACs

In [7], the authors evaluate DeepPROTACs via randomly splitting the PROTAC-DB dataset in 80%-20% percentages. Despite following the same splitting percentages,

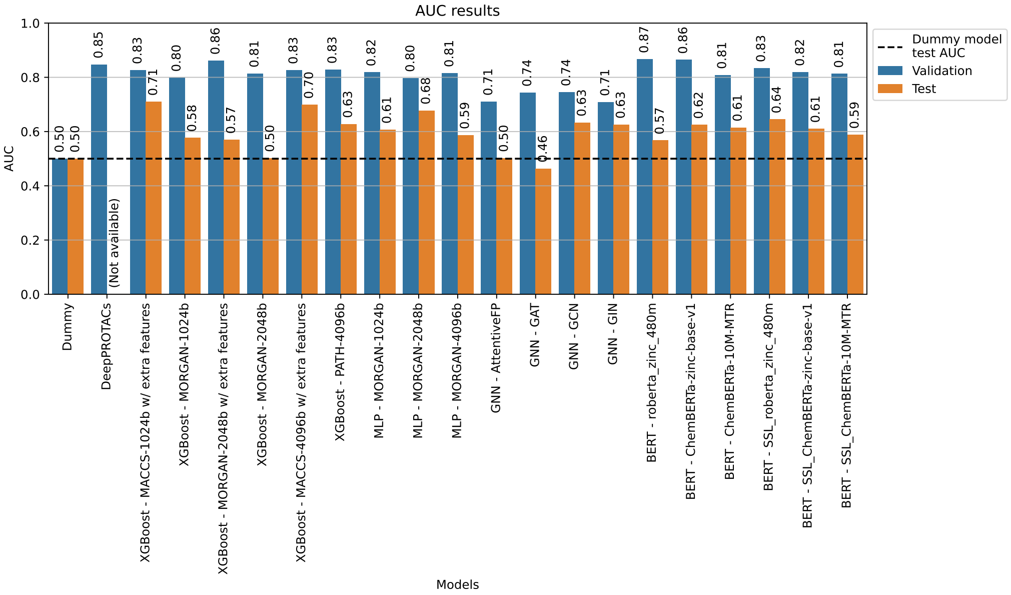


Figure 5.6: AUC scores of the candidate models. The higher the better.

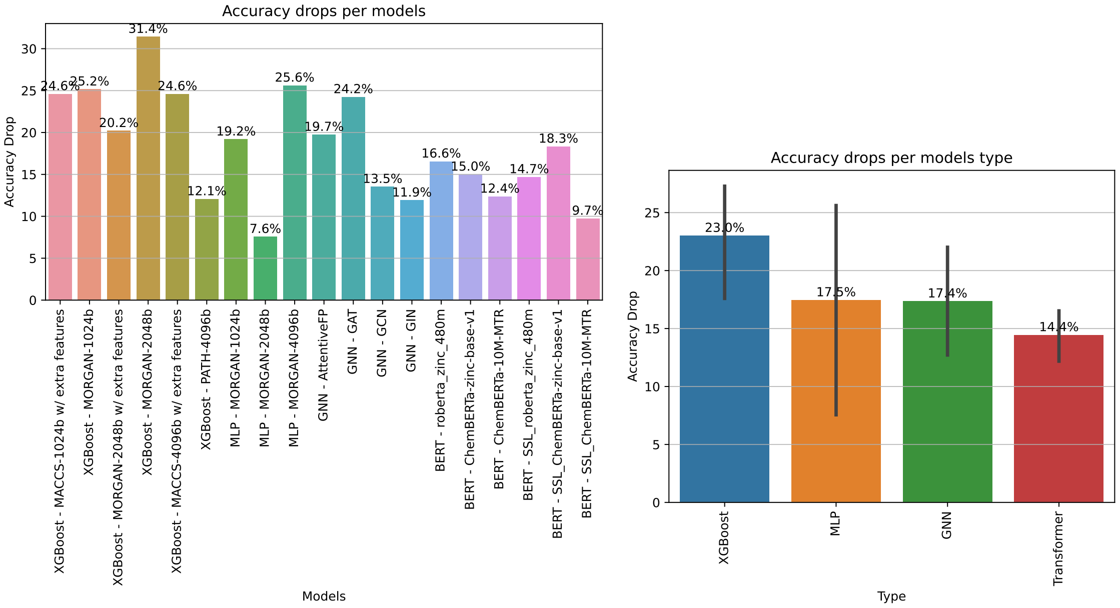
we did not have access to their specific datasets. Moreover, despite being open source, the model is not straightforward to use, since it makes use of advanced and time consuming molecular docking outputs as input features. Thus, we did not manage to evaluate DeepPROTAC on our test set based on PROTAC-Pedia. Hence, we reported its validation results values from their publication, while omitting its test performance.

The validation performance of DeepPROTACs was relatively lower compared to the other models in the dataset. DeepPROTACs achieved a validation accuracy of 77.95% and a validation ROC AUC of 0.847, which are lower than our highest performing models by 3.6% and 0.02 in terms of these metrics, respectively. This indicates that our models, possibly due to their specific architectures, feature representations, or training methodologies, were more effective in capturing and utilizing the relevant patterns and characteristics of the PROTAC complexes despite the limited amount of data available.

Overall, the lower validation performance of DeepPROTACs could indicate potential limitations in its approach or suggest that further optimization or simplifications are required to improve its performance on this particular dataset.

5.4.2 Performance Drops from Validation to Test Sets

The results indicate that several models may have experienced overfitting on the limited data available in the training dataset. Overfitting occurs when a model becomes too complex or overly specialized to the training data, leading to reduced performance on unseen data. One indicator of potential overfitting is when there is a significant difference between the performance on the training/validation data and the performance on the test data. In our case, in Figure 5.7 we observe large drops in accuracy and ROC AUC from the validation set to the test set, suggesting that the models may have learned to perform well on the specific patterns and characteristics of the training/validation data in PROTAC-DB, but failed to generalize to new, unseen examples in PROTAC-Pedia.



(a) (b)

Figure 5.7: Accuracy drops from validation to test sets in the proposed models. The lower the better. (a) Accuracy drops per models. (b) Accuracy drops grouped by model type.

From Figure 5.7b, the XGBoost models exhibit considerably validation-test accuracy drops (23.0% on average). Some models, such as XGBoost - PATH-4096b, showed relatively small drops (12.1% on accuracy), indicating a reasonable level of generalization. However, other XGBoost models, like XGBoost - MORGAN-2048b, experienced larger drops (31.4% on accuracy), suggesting a more pronounced overfitting issue. In general, XGBoost tends to work well when the dataset has limited training data points like in our setup. Deep learning models often require a large amount of data to train effectively, as they are capable of learning complex representations and patterns. In contrast, XGBoost can perform well even with a relatively smaller training dataset due to its ensemble learning nature and the strength of decision trees in capturing relevant features.

Conversely , MLP models demonstrated mixed results in terms of validation-test accuracy drops. While MLP - MORGAN-2048b exhibited a relatively small drop (7.6%), indicating better generalization, MLP - MORGAN-1024b showed a considerable drop (25.6%), implying a higher likelihood of overfitting.

Furthermore, GNN models, including AttentiveFP, GAT, GCN, and GIN, generally experienced significant validation-test accuracy drops (17.4% on average). This suggests that these models may have learned to fit the training/validation data closely, but struggled to generalize their predictions to unseen PROTAC complexes.

On the other hand, Transformer models demonstrated relatively consistent validation-test accuracy drops (14.4% on average). For instance, BERT - Roberta_zinc_480m experienced a drop of 16.6%, while BERT - ChemBERTa-zinc-base-v1 had a drop of 15.0%. These results indicate that the Transformer models maintained a similar level of performance when applied to unseen data, suggesting a relatively good generalization capability. On top of that, the model that was finetuned after self-supervised learning (SSL), SSL_ChemBERTa-10M-MTR, demonstrated the smallest validation-test accuracy drop (9.7%) compared to other Transformer models in the evaluation. This indicates that the SSL pretraining process might have contributed to enhancing the model's ability to generalize to unseen data. In general, however, we see that, compared to non-SSL finetuned models, SSL training does not guarantee high performance in neither accuracy nor AUC scores.

5.4.3 Effect of Hyperparameters

In this Section, we discuss how different hyperparameters could affect the candidate models performance.

5.4.3.1 Fingerprints Bitwidth and Extra Features on XGBoost

As reported in Appendix A, Table A.1, for XGBoost-based models, we wanted to experiment how different fingerprint bitwidths and the presence or absence of "extra features", which refer to additional information such as POI amino acid sequence, cell type, and E3 ligase type, which are incorporated as input features alongside the molecular fingerprints. In particular, for each combination, we allowed Optuna to search and optimize the type of fingerprints to use, being it either Morgan, MACCS or path-based.

Generally, increasing the bitwidth allows for more detailed representation of molecular features, potentially capturing more intricate patterns in the data. This can lead to improved accuracy and AUC scores. However, our analysis indicates that the impact of bitwidth on XGBoost performance is not straightforward. While some higher bitwidths (*e.g.*, 4096) achieved relatively higher accuracy and AUC scores, lower bitwidths (*e.g.*, 1024) also demonstrated competitive performance. Regarding the use "extra features" in XGBoost, we see that their inclusion can have a positive impact on the accuracy and AUC scores. By incorporating contextual information related to the POI, the cellular environment, and the specific E3 ligase type, the model can potentially capture more relevant information for making predictions. This can lead to improved performance compared to using only the molecular fingerprints. Finally, for XGBoost MACCS fingerprints generally yielded higher accuracy and AUC scores (81.6% accuracy and 0.83 AUC on the validation set). The superior performance of MACCS fingerprints suggests that the structural characteristics captured by MACCS are more informative compared to Morgan and path-based fingerprints.

5.4.3.2 Fingerprints Bitwidth on MLP Models

Increasing the fingerprint bitwidth generally allows for a more detailed representation of molecular features, enabling the model to capture finer patterns in the data. Our analysis reveals that higher bitwidths, such as 2048 and 4096, achieved relatively higher accuracy and AUC scores compared to the lower bitwidth of 1024. This indicates that a larger bitwidth provides more information and potentially improves the model's ability to discriminate between different PROTAC molecules, leading to higher performance. In addition to this, the selection of the MORGAN fingerprint type as the best-performing type by Optuna suggests its effectiveness in capturing local and global structural features, leading to improved model performance.

5.4.3.3 Architecture Type on GNN Models

The absence of a specific GNN architecture that outperformed the others suggests that the performance of GNNs in this scenario might be more dependent on factors beyond the architecture itself. We speculate that the comparatively worse performance of all candidate GNNs could be attributed to potentially limited data availability for training compared to fine-tuned BERT models, as well as the high complexity of the selected GNN architectures, which can make them more prone to overfitting. Additionally, since typical PROTAC molecules are fairly long, the limited number of message passing layers in the GNNs could have prevented them from learning from distant nodes in the molecular graphs.

6

Conclusion

This document described our research work on developing a DL-based system for predicting the degradation activity of PROTACs, a class of small molecules for targeted protein degradation.

In Chapter 4 we presented the curation process of the PROTAC-DB dataset. Chapter 4 also reports the proposed model architectures and the experimental setup. In Chapter 5 we showed the results of the methodology described in Chapter 4, which includes an analysis of the curated datasets, together with the accuracy and AUC scores of the candidate models. Chapter 5 ends with a comprehensive discussion on the obtained results, including a comparison with DeepPROTACs and the effects of a limited training dataset on the models performance.

In conclusion, the XGBoost algorithm demonstrated superior performance compared to the DL models (MLP, GNN, Transformer) on the given PROTAC-DB dataset. This could be attributed to XGBoost's ability to effectively handle tabular data, like binary vectors, and its capacity to perform well with limited training data points. Conversely, the DL models, with their higher complexity and greater parameter count, have struggled to generalize effectively due to the limited training data available, but still showed competing performance with respect to XGBoost. Our work also proves the limitations of DeepPROTACs [7] and its lower performance compared to our models. Regarding the test dataset based on PROTAC-Pedia, no clear best model emerged, further highlighting the difficulty of learning PROTACs degradation activity when lacking large amounts of usable training data. We believe that further exploration of data augmentation techniques, regularization methods, and acquiring additional labeled data could potentially enhance the performance of the proposed DL models.

6.1 Future Work

This section reports future ideas and approaches for possible continuations of this thesis work.

6.1.1 Tackling the Problem of Few Data Points

As discussed in Chapter 5, the limited amount of data greatly influenced and affected the ability of the proposed models to generalise to unseen data. In fact, DL models are generally data-hungry models, which struggle to learn complex features representations from few training data. Because of that, one of the most effective way of improving model generalization ability would be to apply data-centric approaches, as described in the following sub-sections.

6.1.1.1 Interpolating Data on the Dose-Response Curve

Around 300 entries of the PROTAC-DB dataset do not explicitly include the exact DC_{50} value, but instead report experimental data points on the dose-response curve, as mentioned in Chapter 4, Section 4.1. In order to estimate the DC_{50} value, one could interpolate such points via an Hill equation, provided there are sufficient data points per entry (in PROTAC-DB, they range from one to four points per entry). However, there are possible limitations on the interpolated data. To start, there is no way to know the true DC_{50} value, so such entries shall not be used for validation nor testing models. Additionally,

6.1.1.2 Semi-Supervised and Active Learning

Semi-supervised learning aims at training machine learning models by using small datasets of labelled data and a large ones of unlabelled data [39], [40]. One of the assumptions behind semi-supervised learning is that data with same, or very similar, labels are close in high-density regions of the feature space. Hence, by leveraging correlations and similarities between labelled and unlabelled data points one could potentially still learn meaningful data representations and therefore train an ML model.

Another possible way to exploit unlabelled data would be to apply *active learning* techniques [41], [42]. For instance, an oracle model pretrained on the labeled dataset can be used to label unlabeled data. The newly labeled data that reach a high confidence score are then included into the training set. Such updated training set can then be used to train a new model to repeat the labeling on the remaining unlabelled points.

6.1.1.3 Transfer Learning

Contrary to PROTAC-DB, the PROTAC-Pedia dataset presents many more active/inactive entries. However, PROTAC-Pedia definition of an active PROTAC is less strict compared to PROTAC-DB, being active when $D_{max} \geq 30\%$ and $DC_{50} \leq 10\mu M$. A possible way to train a model to predict PROTAC-DB definition of active/inactive entries ($D_{max} \geq 80\%$ and $DC_{50} \leq 0.1\mu M$) would then be to train a model on PROTAC-Pedia entries and then finetune the same model on PROTAC-DB. This will effectively increase the amount of training data points, but might still not be sufficient for avoiding overfitting on the few available data of PROTAC-DB.

6.1.1.4 Advanced Data Analysis

Principal Component Analysis (PCA) [43] and t-Distributed Stochastic Neighbor Embedding (t-SNE) [44] are two popular dimensionality reduction techniques that can provide valuable insights about data. In future work, such techniques can be applied to different molecular and PROTAC complex representations in order to extract deeper insights from the available data. For example, clustering algorithms can be applied to the reduced feature space obtained from t-SNE to identify distinct groups within the data or within specific datasets. Furthermore, correlation analysis can be performed on the principal components generated by PCA to uncover relationships between features. Additionally, feature selection techniques can be employed using feature importance scores obtained from PCA to identify the most relevant variables for modeling or further analysis.

6.1.2 More Advanced Feature Representations

Due to time constraints, we focused our research work on the most promising and popular PROTAC-POI-E3 ligase complex representations. There are however many possible ways to encode more advanced representations into our candidate deep learning model.

6.1.2.1 More Molecular Representations

As mentioned in Chapter 2, Section 2.2.1, many molecular representations have been proposed in literature. For instance, Winter *et al.* [45] propose a new method for learning and generating Continuous and Data-Driven molecular Descriptors (CDDD). Their methodology is based on the idea of translating equivalent chemical representations into a continuous space. The authors first translate the molecules into a set of equivalent chemical representations. This can be done via SMILES strings, InChI strings, 3D structures, *et cetera.* Once the molecules have been translated, they are then embedded into a common latent space. This is done using a neural network (a variational autoencoder [46]), which learns to map the molecules from their original representations to the latent space. The descriptors are finally learned by projecting the molecules from the latent space back to their original representations. This is done using a linear projection, which is trained to minimize the reconstruction error. A possible future work direction could be designing a SMILES encoder that leverage CDDD vectors extracted from PROTAC SMILES and compare it to other molecular representations.

6.1.2.2 3D Information from the PROTAC-POI-E3 Ligase Complex

By incorporating 3D information, we could leverage the structural characteristics of PROTACs and their target proteins to enhance the accuracy and performance of machine learning models for predicting degradation activity. This can help capture critical features related to binding affinity, shape complementarity, and specific protein-ligand interactions that can influence degradation activity. Future work could explore different techniques for modelling and encoding 3D information. For example, the PRosettaC software [47] is able to accurately predict the binding affinities of PROTACs to their targets, as well as the efficiency of the PROTACs in inducing target degradation. It does so by modeling the formation of ternary complexes between PROTACs, protein targets, and E3 ligases. These models could eventually be leveraged as additional input features to more advanced machine learning models. Potentially, 3D simulations could also be learned from bare PROTAC SMILES and amino acid sequences of the POI and E3 ligase, perhaps via exploiting and modifying layer modules from the Alpha- and Open-Fold projects [48], [49].

6.1.3 Predicting Protein Degradation as a Regression Task

Since the degradation activity of PROTACs is not unanimously defined, it might useful to directly predict the D_{max} and DC_{50} values of a given PROTAC molecule. This, however, introduces additional challenges in designing an accurate machine learning system for this task. The D_{max} and DC_{50} values reported in PROTAC-DB and PROTAC-Pedia are themselves interpolated values on the dose-response curve, whose approximation highly depend on the amount and quality of the data points measured during laboratory assays (refer to Figure 2.2a). As such, D_{max} and DC_{50} values alone might be far from an accurate description of PROTACs behavior. Hence, it might be extremely challenging for a model to learn to predict them. On top of that, we are currently facing an overfitting problem due to the low amount of data. We speculate that for training on a regression task, the required number of data points might increase, becoming too high to overcome for the current state of the available open PROTACs datasets.

6.1.4 Explainability

Deep learning models are generally treated as black-boxes, meaning that their inner parameters are hard to interpret and the layers input-output relationship are difficult to comprehend. Despite this, in recent years there has been an increasing interest in understanding models parameters and explaining their role in generating the models' outputs [50]. In fact, knowing more about how a model reacts to different inputs can become beneficial for identifying weaknesses in the model itself and/or gain more insights about the data it processes. In future work, advanced visualization tools like SHAP [51] can be applied to trained models in order to get more information about their parameters in relation to the data features.

6.1.4.1 Feature Importance

Feature importance is a technique used in XGBoost and other machine learning algorithms to measure the relative importance of each data feature in predicting the target variable. To do so, it assigns a score to each feature based on its contribution to the model's performance. The higher the score, the more important the feature is in making predictions. Thanks to this score, our proposed trained XGBoost models can be inspected in order to identify which bit positions in the fingerprints are the most relevant for predicting PROTAC degradation activity. In turn, such bit can ideally tell us which molecular characteristics are the most informative for training machine learning models. However, fingerprints might have single bits encoding information from multiple characteristics, as described in Chapter 2, Section 2.2.1. Because of that, it is not straightforward to reverse engineer feature importance scores.

Bibliography

[1] Z. Hu and C. M. Crews, "Recent Developments in PROTAC-Mediated Protein Degradation: From Bench to Clinic," *ChemBioChem*, vol. 23, no. 2, e202100270, 2022.

[2] S. Tomoshige and M. Ishikawa, "PROTACs and other chemical protein degradation technologies for the treatment of neurodegenerative disorders," *Angewandte Chemie International Edition*, vol. 60, no. 7, pp. 3346–3354, 2021.

[3] J. Liu, J. Ma, Y. Liu, *et al.*, "PROTACs: a novel strategy for cancer therapy," in *Seminars in Cancer Biology*, Elsevier, vol. 67, 2020, pp. 171–179.

[4] D. Nori, C. W. Coley, and R. Mercado, "De novo PROTAC design using graph-based deep generative models," *arXiv preprint arXiv:2211.02660*, 2022.

[5] G. Weng, C. Shen, D. Cao, *et al.*, "PROTAC-DB: an online database of PROTACs," *Nucleic acids research*, vol. 49, no. D1, pp. D1381–D1387, 2021.

[6] *PROTACpedia - Main.* [Online]. Available: `https://protacpedia.weizmann.ac.il/ptcb/main` (visited on 05/26/2023).

[7] F. Li, Q. Hu, X. Zhang, *et al.*, "DeepPROTACs is a deep learning-based targeted degradation predictor for PROTACs," *Nature Communications*, vol. 13, no. 1, p. 7133, Nov. 21, 2022, Number: 1 Publisher: Nature Publishing Group, ISSN: 2041-1723. DOI: `10.1038/s41467-022-34807-3`. [Online]. Available: `https://www.nature.com/articles/s41467-022-34807-3` (visited on 04/03/2023).

[8] R. Gesztelyi, J. Zsuga, A. Kemeny-Beke, B. Varga, B. Juhasz, and A. Tosaki, "The Hill equation and the origin of quantitative pharmacology," en, *Archive for History of Exact Sciences*, vol. 66, no. 4, pp. 427–438, Jul. 2012, ISSN: 1432-0657. DOI: `10.1007/s00407-012-0098-5`. [Online]. Available: `https://doi.org/10.1007/s00407-012-0098-5` (visited on 05/03/2023).

[9] E. Semenova, M. L. Guerriero, B. Zhang, *et al.*, "Flexible Fitting of PROTAC Concentration-Response Curves with Changepoint Gaussian Processes," eng, *SLAS discovery: advancing life sciences R & D*, vol. 26, no. 9, pp. 1212–1224, Oct. 2021, ISSN: 2472-5560. DOI: `10.1177/24725552211028142`.

[10] H. L. Morgan, "The Generation of a Unique Machine Description for Chemical Structures-A Technique Developed at Chemical Abstracts Service.," *Journal of Chemical Documentation*, vol. 5, no. 2, pp. 107–113, May 1965, Publisher: American Chemical Society, ISSN: 0021-9576. DOI: `10.1021/c160017a018`. [Online]. Available: `https://doi.org/10.1021/c160017a018` (visited on 02/10/2023).

[11] J. L. Durant, B. A. Leland, D. R. Henry, and J. G. Nourse, "Reoptimization of MDL keys for use in drug discovery," eng, *Journal of Chemical Information*

and Computer Sciences, vol. 42, no. 6, pp. 1273–1280, 2002, ISSN: 0095-2338. DOI: `10.1021/ci010132r`.

[12] *Rdkit.Chem.rdmolops module — The RDKit 2023.03.1 documentation.* [Online]. Available: `https://www.rdkit.org/docs/source/rdkit.Chem.rdmolops.html` (visited on 05/03/2023).

[13] D. Baptista, J. Correia, B. Pereira, and M. Rocha, "Evaluating molecular representations in machine learning models for drug response prediction and interpretability," en, *Journal of Integrative Bioinformatics*, vol. 19, no. 3, Sep. 2022, Publisher: De Gruyter, ISSN: 1613-4516. DOI: `10.1515/jib-2022-0006`. [Online]. Available: `https://www.degruyter.com/document/doi/10.1515/jib-2022-0006/html` (visited on 05/04/2023).

[14] D. Weininger, "SMILES, a chemical language and information system. 1. Introduction to methodology and encoding rules," *Journal of Chemical Information and Computer Sciences*, vol. 28, no. 1, pp. 31–36, Feb. 1988, Publisher: American Chemical Society, ISSN: 0095-2338. DOI: `10.1021/ci00057a005`. [Online]. Available: `https://doi.org/10.1021/ci00057a005` (visited on 05/04/2023).

[15] K. Fukushima, "Cognitron: A self-organizing multilayered neural network," eng, *Biological Cybernetics*, vol. 20, no. 3-4, pp. 121–136, Nov. 1975, ISSN: 0340-1200. DOI: `10.1007/BF00342633`.

[16] Y. LeCun, P. Haffner, L. Bottou, and Y. Bengio, "Object Recognition with Gradient-Based Learning," en, in *Shape, Contour and Grouping in Computer Vision*, ser. Lecture Notes in Computer Science, D. A. Forsyth, J. L. Mundy, V. di Gesú, and R. Cipolla, Eds., Berlin, Heidelberg: Springer, 1999, pp. 319–345, ISBN: 978-3-540-46805-9. DOI: `10.1007/3-540-46805-6_19`. [Online]. Available: `https://doi.org/10.1007/3-540-46805-6_19` (visited on 05/04/2023).

[17] T. N. Kipf and M. Welling, *Semi-Supervised Classification with Graph Convolutional Networks*, Number: arXiv:1609.02907 arXiv:1609.02907 [cs, stat], Feb. 2017. DOI: `10.48550/arXiv.1609.02907`. [Online]. Available: `http://arxiv.org/abs/1609.02907` (visited on 06/02/2023).

[18] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, *Graph Attention Networks*, Number: arXiv:1710.10903 arXiv:1710.10903 [cs, stat], Feb. 2018. DOI: `10.48550/arXiv.1710.10903`. [Online]. Available: `http://arxiv.org/abs/1710.10903` (visited on 06/02/2023).

[19] J. S. Bridle, "Training stochastic model recognition algorithms as networks can lead to maximum mutual information estimation of parameters," in *Proceedings of the 2nd International Conference on Neural Information Processing Systems*, ser. NIPS'89, Cambridge, MA, USA: MIT Press, Jan. 1989, pp. 211–217. (visited on 05/04/2023).

[20] B. Xu, N. Wang, T. Chen, and M. Li, *Empirical Evaluation of Rectified Activations in Convolutional Network*, arXiv:1505.00853 [cs, stat], Nov. 2015. DOI: `10.48550/arXiv.1505.00853`. [Online]. Available: `http://arxiv.org/abs/1505.00853` (visited on 06/07/2023).

[21] A. Vaswani, N. Shazeer, N. Parmar, *et al.*, *Attention Is All You Need*, Number: arXiv:1706.03762 arXiv:1706.03762 [cs], Dec. 2017. DOI: `10.48550/arXiv.`

1706.03762. [Online]. Available: http://arxiv.org/abs/1706.03762 (visited on 05/03/2023).

[22] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*, Number: arXiv:1810.04805 arXiv:1810.04805 [cs], May 2019. DOI: 10.48550/arXiv.1810.04805. [Online]. Available: http://arxiv.org/abs/1810.04805 (visited on 05/03/2023).

[23] C. Knutson, M. Bontha, J. A. Bilbrey, and N. Kumar, "Decoding the protein-ligand interactions using parallel graph neural networks," *Scientific reports*, vol. 12, no. 1, pp. 1–14, 2022.

[24] M. A. Moesser, D. Klein, F. Boyles, C. M. Deane, A. Baxter, and G. M. Morris, "Protein-Ligand Interaction Graphs: Learning from Ligand-Shaped 3D Interaction Graphs to Improve Binding Affinity Prediction," *bioRxiv*, 2022.

[25] J. Lee, W. Yoon, S. Kim, *et al.*, "BioBERT: A pre-trained biomedical language representation model for biomedical text mining," *Bioinformatics*, vol. 36, no. 4, pp. 1234–1240, Feb. 2020, arXiv:1901.08746 [cs], ISSN: 1367-4803, 1367-4811. DOI: 10.1093/bioinformatics/btz682. [Online]. Available: http://arxiv.org/abs/1901.08746 (visited on 05/03/2023).

[26] *The RDKit Documentation — The RDKit 2023.03.1 documentation.* [Online]. Available: https://www.rdkit.org/docs/index.html (visited on 05/03/2023).

[27] *PyG Documentation — pytorch_geometric documentation.* [Online]. Available: https://pytorch-geometric.readthedocs.io/en/stable/ (visited on 05/03/2023).

[28] S. Wang, Y. Guo, Y. Wang, H. Sun, and J. Huang, "SMILES-BERT: Large Scale Unsupervised Pre-Training for Molecular Property Prediction," in *Proceedings of the 10th ACM International Conference on Bioinformatics, Computational Biology and Health Informatics*, ser. BCB '19, New York, NY, USA: Association for Computing Machinery, Sep. 2019, pp. 429–436, ISBN: 978-1-4503-6666-3. DOI: 10.1145/3307339.3342186. [Online]. Available: https://doi.org/10.1145/3307339.3342186 (visited on 05/04/2023).

[29] L. Ericsson, H. Gouk, C. C. Loy, and T. M. Hospedales, "Self-Supervised Representation Learning: Introduction, Advances and Challenges," *IEEE Signal Processing Magazine*, vol. 39, no. 3, pp. 42–62, May 2022, arXiv:2110.09327 [cs, stat], ISSN: 1053-5888, 1558-0792. DOI: 10.1109/MSP.2021.3134634. [Online]. Available: http://arxiv.org/abs/2110.09327 (visited on 06/02/2023).

[30] *Optuna: A hyperparameter optimization framework — Optuna 3.2.0 documentation.* [Online]. Available: https://optuna.readthedocs.io/en/stable/index.html (visited on 06/02/2023).

[31] *XGBoost Documentation — xgboost 1.7.5 documentation.* [Online]. Available: https://xgboost.readthedocs.io/en/stable/index.html (visited on 06/02/2023).

[32] T. Chen, T. He, M. Benesty, *et al.*, "XGBoost: Extreme gradient boosting," *R package version 0.4-2*, vol. 1, no. 4, pp. 1–4, 2015.

[33] *Pushing the Boundaries of Molecular Representation for Drug Discovery with the Graph Attention Mechanism | Journal of Medicinal Chemistry.* [Online].

Available: https://pubs.acs.org/doi/10.1021/acs.jmedchem.9b00959 (visited on 06/02/2023).

[34] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, *How Powerful are Graph Neural Networks?* Number: arXiv:1810.00826 arXiv:1810.00826 [cs, stat], Feb. 2019. DOI: 10.48550/arXiv.1810.00826. [Online]. Available: http://arxiv.org/abs/1810.00826 (visited on 06/02/2023).

[35] *Entropy/roberta_zinc_480m · Hugging Face.* [Online]. Available: https://huggingface.co/entropy/roberta_zinc_480m (visited on 06/07/2023).

[36] W. Ahmad, E. Simon, S. Chithrananda, G. Grand, and B. Ramsundar, *ChemBERTa-2: Towards Chemical Foundation Models*, arXiv:2209.01712 [cs, q-bio], Sep. 2022. DOI: 10.48550/arXiv.2209.01712. [Online]. Available: http://arxiv.org/abs/2209.01712 (visited on 06/07/2023).

[37] B. Ramsundar, P. Eastman, P. Walters, V. Pande, K. Leswing, and Z. Wu, *Deep Learning for the Life Sciences.* O'Reilly Media, 2019, https://www.amazon.com/Deep-Learning-Life-Sciences-Microscopy/dp/1492039837.

[38] *Hugging Face – The AI community building the future.* May 2023. [Online]. Available: https://huggingface.co/ (visited on 06/02/2023).

[39] L. Weng, *Learning with not Enough Data Part 1: Semi-Supervised Learning*, en, Section: posts, Dec. 2021. [Online]. Available: https://lilianweng.github.io/posts/2021-12-05-semi-supervised/ (visited on 06/15/2023).

[40] Y. Ouali, C. Hudelot, and M. Tami, *An Overview of Deep Semi-Supervised Learning*, arXiv:2006.05278 [cs, stat], Jul. 2020. DOI: 10.48550/arXiv.2006.05278. [Online]. Available: http://arxiv.org/abs/2006.05278 (visited on 06/15/2023).

[41] P. Ren, Y. Xiao, X. Chang, *et al.*, "A survey of deep active learning," *ACM computing surveys (CSUR)*, vol. 54, no. 9, pp. 1–40, 2021.

[42] B. Settles, "Active learning literature survey," 2009.

[43] H. Abdi and L. J. Williams, "Principal component analysis," *Wiley interdisciplinary reviews: computational statistics*, vol. 2, no. 4, pp. 433–459, 2010.

[44] L. van der Maaten and G. Hinton, "Viualizing data using t-SNE," *Journal of Machine Learning Research*, vol. 9, pp. 2579–2605, Nov. 2008.

[45] R. Winter, F. Montanari, F. Noé, and D.-A. Clevert, "Learning continuous and data-driven molecular descriptors by translating equivalent chemical representations †Electronic supplementary information (ESI) available: Detailed information regarding the final model architecture, hyperparameter grid, results and computation time. See DOI: 10.1039/c8sc04175j," *Chemical Science*, vol. 10, no. 6, pp. 1692–1701, Nov. 2018, ISSN: 2041-6520. DOI: 10.1039/c8sc04175j. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6368215/ (visited on 06/14/2023).

[46] D. P. Kingma and M. Welling, *Auto-Encoding Variational Bayes*, arXiv:1312.6114 [cs, stat], Dec. 2022. DOI: 10.48550/arXiv.1312.6114. [Online]. Available: http://arxiv.org/abs/1312.6114 (visited on 06/14/2023).

[47] D. Zaidman, J. Prilusky, and N. London, "PRosettaC: Rosetta Based Modeling of PROTAC Mediated Ternary Complexes," *Journal of Chemical Information and Modeling*, vol. 60, no. 10, pp. 4894–4903, Oct. 2020, Publisher: American Chemical Society, ISSN: 1549-9596. DOI: 10.1021/acs.jcim.0c00589. [Online].

Available: https://doi.org/10.1021/acs.jcim.0c00589 (visited on 06/14/2023).

[48] J. Jumper, R. Evans, A. Pritzel, *et al.*, “Highly accurate protein structure prediction with AlphaFold,” en, *Nature*, vol. 596, no. 7873, pp. 583–589, Aug. 2021, Number: 7873 Publisher: Nature Publishing Group, ISSN: 1476-4687. DOI: 10.1038/s41586-021-03819-2. [Online]. Available: https://www.nature.com/articles/s41586-021-03819-2 (visited on 06/14/2023).

[49] G. Ahdritz, N. Bouatta, S. Kadyan, *et al.*, *OpenFold: Retraining AlphaFold2 yields new insights into its learning mechanisms and capacity for generalization*, en, Nov. 2022. DOI: 10.1101/2022.11.20.517210. [Online]. Available: https://www.biorxiv.org/content/10.1101/2022.11.20.517210v2 (visited on 02/07/2023).

[50] N. Burkart and M. F. Huber, “A survey on the explainability of supervised machine learning,” *Journal of Artificial Intelligence Research*, vol. 70, pp. 245–317, 2021.

[51] *Welcome to the SHAP documentation — SHAP latest documentation.* [Online]. Available: https://shap.readthedocs.io/en/latest/ (visited on 06/14/2023).

A

Hyperparameters Search Space

Hyperparameter optimization plays a crucial role in machine learning and model development. Fine-tuning the hyperparameters can significantly impact the performance and generalization of a model. In this Appendix chapter, we present the hyperparameters search space of the different candidate models we proposed in our work. In the following tables, hyperparameters not marked as "fixed" have been optimized via Optuna. In particular, values within curly brackets {} represent categorical choices, whereas those in square brackets [ ] report a range of possible values to be selected.

Table A.1 summarizes the hyperparameters for the XGBoost-based baseline model. It includes parameters such as the type of fingerprint used, *e.g.*, MORGAN fingerprint, MACCS fingerprint, path fingerprint, fingerprint radius, and maximum path length. Additionally, XGBoost-specific parameters such as the booster type, regularization parameters (lambda and alpha), and growth policies are presented.

Table A.2 provides an overview of the hyperparameters for the candidate model with MLP-based SMILES encoder. It includes parameters such as the number of layers, hidden channels, and dropout rates for the SMILES encoder.

The hyperparameters for the GNN model are presented in Table A.3. It includes the choice of GNN type (like "gin" and "attentivefp"), number of layers, dropout rates, and conditional parameters such as output channels and JK (Jumping Knowledge) mechanisms.

Finally, Table A.4 outlines the hyperparameters for the BERT-based model. It includes parameters such as the number of layers, hidden channels, and dropout rates for the extra features branch, together with learning rate, batch size, and accumulate gradient batches.

Table A.1: XGBoost models hyperparameters.

Hyperparameter	**Range**
Fingerprint-specific	
fp_bits	{1024, 2048, 4096} (fixed)
fp_type	{morgan_fp, maccs_fp, path_fp}
fp_radius	[2, 10]
fp_max_path	[8, 10]
XGBoost-specific	
booster	{gbtree, gblinear, dart}
lambda	[1e-8, 1.0] (log scale)
alpha	[1e-8, 1.0] (log scale)
Additional hyperparameters for booster=gbtree or booster=dart	
max_depth	[1, 16]
eta	[1e-8, 1.0] (log scale)
gamma	[1e-8, 1.0] (log scale)
grow_policy	{depthwise, lossguide}
Additional hyperparameters for booster=dart	
sample_type	{uniform, weighted}
normalize_type	{tree, forest}
rate_drop	[1e-8, 1.0] (log scale)
skip_drop	[1e-8, 1.0] (log scale)
Additional hyperparameters (conditional)	
fp_use_extra_features	{True, False}

Table A.2: Hyperparameters of MLP-based SMILES encoder models.

Hyperparameter	**Range**
num_epochs	(fixed value)
num_layers_extra	[2, 8]
hidden_channels_extra_features	[32, 1024] (step=32)
dropout	[0.1, 0.8]
learning_rate	[1e-5, 1e-2] (log scale)
batch_size	{16, 32, 64, 128}
SMILES Encoder-specific	
fp_bits	{1024, 2048, 4096} (fixed)
fp_type	{morgan_fp, maccs_fp, path_fp}
fp_radius	[2, 10] (for MORGAN only)
fp_max_path	[8, 10] (for path-based only)
num_layers	[2, 8]
hidden_channels	[32, 1024] (step=32)
dropout	[0.1, 0.8]

Table A.3: Hyperparameters of GNN-based SMILES encoder models.

Hyperparameter	**Range**
num_epochs	(fixed value)
num_layers_extra	[2, 8]
hidden_channels_extra_features	[64, 512] (step=32)
dropout	[0.01, 0.8]
learning_rate	[1e-5, 1e-2] (log scale)
batch_size	{4, 8}
SMILES Encoder-specific	
gnn_type	{gin, gat, gcn, attentivefp} (fixed)
hidden_channels	[64, 768] (step=64)
num_layers	[2, 8]
dropout	[0.01, 0.8]
out_channels	{64, 128, 256, 512, 768}
num_timesteps	{8, 16, 32, 64, 128, 256, 512, 768}
jk	{max, last, cat, lstm}

Table A.4: Hyperparameters of Transformer-based SMILES encoder models.

Hyperparameter	**Range**
num_epochs	(fixed value)
num_layers_extra	[2, 8]
hidden_channels_extra_features	[64, 512] (step=32)
dropout	[0.01, 0.8]
learning_rate	[1e-5, 1e-2] (log scale)
batch_size	{4, 8}
SMILES Encoder-specific	
bert_model	(fixed value)
accumulate_grad_batches	{1, 2, 4, 8}