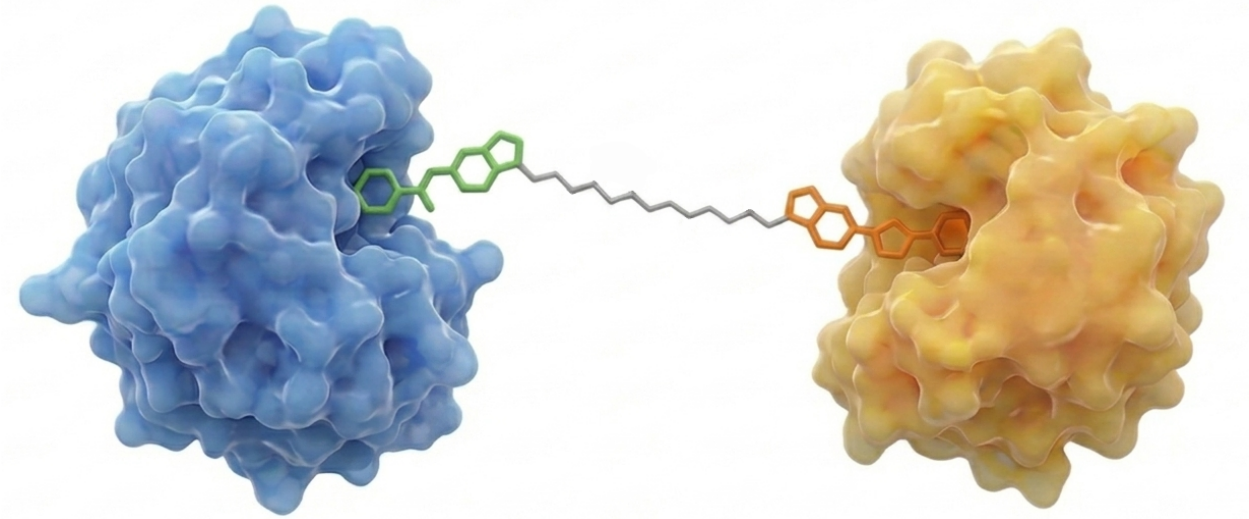
CHALMERS
UNIVERSITY OF TECHNOLOGY



Reinforcement Learning and Generative AI for Next-Generation Protein Degradation Drugs

Master's thesis in Computer Science and Engineering

ALEXANDER PERSSON
FELIX ERNGÅRD

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2026
www.chalmers.se

Master's thesis 2026

Reinforcement Learning and Generative AI for Next-Generation Protein Degradation Drugs

ALEXANDER PERSSON
FELIX ERNGÅRD

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Computer Science and Engineering
Division of Data Science and AI
AI Laboratory for Molecular Engineering (AIME)
Chalmers University of Technology
Gothenburg, Sweden 2026

Reinforcement Learning and Generative AI for Next-Generation Protein Degradation Drugs
ALEXANDER PERSSON
FELIX ERNGÅRD

Supervisor: Stefano Ribes, Data Science and AI, Department of Computer Science and Engineering
Examiner: Assist. Prof. Rocío Mercado, Data Science and AI, Department of Computer Science and Engineering

Master's Thesis 2026
Department of Computer Science and Engineering
Division of Data Science and AI
AI Laboratory for Molecular Engineering (AIME)
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: Illustration of a PROTAC connecting a POI and an E3 ligase. (AI generated by Google Gemini 3.5 Flash)

Typeset in LaTeX
Printed by Chalmers Reproservice
Gothenburg, Sweden 2026

Reinforcement Learning and Generative AI for Next-Generation Protein Degradation Drugs

Alexander Persson
Felix Erngård

Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg

Abstract

Targeted protein degradation using Proteolysis Targeting Chimeras (PROTACs) has emerged as a new paradigm in drug discovery, enabling event-driven pharmacology targeting disease-causing proteins previously considered undruggable. However, due to the complex ternary structure of PROTACs comprising a POI warhead, E3 ligand, and a linker, optimizing them for degradation presents a difficult multi-objective optimization task. This is because minor structural changes can significantly impact the degradation potency (DC_{50}) and maximum degradation level (D_{max}). This thesis introduces a pipeline for *de novo* generation of text-based SMILES representations of PROTACs optimized for degradation of the Androgen receptor (AR) using the Cereblon (CRBN) E3 ligase, and the LNCaP cell line. The thesis implements and evaluates two deep learning architectures: a long short-term memory (LSTM)-based recurrent neural network (RNN) trained using the REINVENT4 framework, and an autoregressive Transformer-based model. For both models, the pipeline utilizes multi-stage transfer learning (TL) across synthetic and curated PROTAC data, after which reinforcement learning (RL) is applied, driven by rewards given by the ensemble of TACK surrogate degradation predictors. To rigorously test the generalization capabilities of both models, they were trained using three increasingly restricting datasets: (1) a dataset where data with an exact scaffold match to any PROTAC in a held-out set were removed, (2) identical to (1), with additional removal of data having a Tanimoto similarity > 0.4 to any PROTAC scaffold in the held-out set, and (3) identical to condition (2), with the additional removal of all data associated with the target POI (AR). In addition to the multi-stage training, the pipeline includes a data-processing step and an evaluation step.

Out of 10,000 generated PROTACS, the fully trained LSTM on the largest dataset achieved 97.9% validity and 99.8% novelty. Furthermore, 9222 out of the 10,000 sampled SMILES were valid, unique, and canonical, of which 96.5% had predicted $DC_{50} < 100$ nM, 85.6% had predicted $D_{max} > 80\%$, and 91.7% were predicted to be active degraders for the target POI according to surrogate degradation predictor models. Additionally, the fully trained Transformer trained on the largest dataset achieved 95.6% validity and 100% novelty. 9112 out of the 10,000 sampled SMILES were valid, unique, and canonical, of which 99.4% had predicted $DC_{50} < 100$ nM, 98.6% had predicted $D_{max} > 80\%$, and 94.9% were predicted to be active degraders for the target POI according to surrogate degradation predictor models. These results indicate that generative AI pipelines utilizing reinforcement learning can teach models to discover highly potent and novel PROTACs under multiple complex

objectives.

Keywords: PROTAC, generative AI, drug discovery, targeted protein degradation, machine learning, reinforcement learning, transfer learning, data.

Acknowledgements

We would like to thoroughly thank our supervisor, Stefano Ribes, for your continued support throughout the thesis work. You were always there to help us in all our challenging moments. We thank you for our insightful discussions and for going out of your way to support us whenever we needed. Also, we would like to thank you for the TACK degradation predictor models, which made the thesis possible. Finally, we would like to thank our examiner, Rocío Mercado, and everyone in the AIME team for making us feel welcome and trusting us with this work.

Alexander Persson and Felix Erngård, Gothenburg, 2026-06-12

List of Acronyms

Below is the list of acronyms that have been used throughout this thesis listed in alphabetical order:

AI	Artificial Intelligence
AR	Androgen Receptor
CRBN	Cereblon
DAP	Difference between augmented and posterior
ECFP	Extended-connectivity fingerprint
GRPO	Group relative policy optimization
HBA	Hydrogen bond acceptors
HBD	Hydrogen bond donors
LNCaP	Lymph node carcinoma of the prostate
LSTM	Long short-term memory
MDP	Markov decision process
MOO	Multi-objective optimization
MW	Molecular weight
NLL	Negative log-likelihood
NLP	Natural language processing
nM	nanomolar units
POI	Protein of interest
PPO	Proximal policy optimization
PROTAC	Proteolysis targeting chimera
RL	Reinforcement learning
RNN	Recurrent neural network
Ro5	Lipinski's rule of 5
RoP	Rule of oral PROTACs
RotB	Rotational bonds
SMILES	Simplified molecular input line entry specification
TACK	Targeting chimeras knowledge
TL	Transfer learning
TOPSIS	Technique for order preference by similarity to ideal solution
TPDdb	Targeted protein degradation database
tPSA	Topological polar surface area
UPS	Ubiquitin-proteasome system
V.U.N.	Validity, uniqueness, and novelty

Contents

List of Figures

List of Tables

1

Introduction

Most therapeutics in use today are based on small molecules and occupancy-driven pharmacology [1]. The effect of these conventional inhibitor drugs relies on the sustained occupancy of the active site of the target protein. In other words, rather than destroying the protein of interest (POI), conventional inhibitor drugs render the POI temporarily ineffective, resulting in the need for continuous medication, often in high doses [1]. This requirement for high drug dosage often leads to undesirable side effects due to off-target interactions. In addition, mutations in the POI can reduce drug effectiveness. In recent years, technological advancements in the field of biological chemistry have enabled event-driven pharmacology through targeted protein degradation using PROteolysis TArgeting Chimeras (PROTACs) [2], [3]. The introduction of event-driven pharmacology through PROTACs has shown effectiveness in degrading proteins previously rendered "undruggable" by conventional small molecule drugs [4].

PROTACs are small molecules consisting of three regions: a binding region for the POI (warhead), a binding region for the E3 ligase (E3 ligand), and a linker connecting the warhead and the E3 ligand [5]. Each component plays a vital role in the effectiveness of the PROTAC. The warhead and E3 ligands must allow for proper binding to the POI and the E3 ligase to achieve high degradation efficiency, whereas the linker design is more flexible. However, PROTAC design is a non-trivial task due to the PROTAC's flexibility and the vast chemical space. The concept of molecular design focuses on inferring structural aspects of the molecule based on specified properties [6]. By leveraging a molecular design approach in combination with modern machine learning, novel molecules can be designed by learning a mapping from desired properties to a subset of the chemical space. Thus, limiting the number of potential candidate designs to those with favorable properties.

Numerous measurable properties have been used as targets of optimization for de novo drug design. Zheng *et al.* [7] optimized for pharmacokinetic (PK) properties and Nori *et al.* [8] focused on optimizing for high potency through DC_{50} (concentration which resulted in 50% targeted protein degradation). Additional optimization targets include, but are not limited to, IC_{50} (concentration of the PROTAC required to inhibit a POI by 50%) and D_{max} (maximum level of protein degradation). However, the more optimization targets used, the more difficult the optimization task be-

comes. This requires both time-consuming and complex construction of the reward, which may ultimately lead to suboptimal degradation in favor of other properties.

This thesis aims to investigate whether SMILES-based generative AI models can learn to generate biologically active PROTACs while simultaneously optimizing multiple objectives, specifically DC_{50} and D_{max}. In detail, the thesis aims to answer whether it is possible to generate potent PROTACs for the degradation of the AR protein using the CRBN E3 ligase and the LNCaP cell line, given surrogate degradation predictions. In order to optimize for DC_{50} and D_{max}, reinforcement learning (RL) will be applied, and its effectiveness for guiding PROTAC generation will be evaluated. Furthermore, the thesis aims to investigate if, and how, differences in training data during transfer learning (TL) affect the generalization during RL. This includes training models using: (1) a dataset where data with an exact scaffold match to any PROTAC in a held-out set were removed, (2) identical to (1), with additional removal of data having a Tanimoto similarity > 0.4 to any PROTAC scaffold in the held-out set, and (3) identical to condition (2), with the additional removal of all data associated with the target POI (AR). These settings evaluate the models ability to generalize beyond structurally similar compounds and generate active PROTACs for previously unseen targets. Furthermore, the thesis aims to answer how reinforcement learning (RL) can be implemented to optimize PROTAC generation and its effect on the PROTAC generation capabilities. For all three generalization settings, a model is considered successful if it, after full training, achieves a mean $DC_{50} < 100$ nM, mean $D_{max} > 80\%$, and validity $> 95\%$.

The thesis implies novelty by extending the currently available research by applying a similar approach of RL to Zheng *et al.* [7]. Additionally, the thesis combines this with the idea of generating entire PROTAC sequences as seen in Nori *et al.* [8] while using SMILES-based models instead of graph-based. Furthermore, the thesis adds novelty by optimizing for both DC_{50} and D_{max} simultaneously using the TACK [9] degradation predictor models. In contrast, Nori *et al.* optimized solely for DC_{50}, and Zheng *et al.* optimized for linker length, PK properties, and LogP. Finally, the thesis utilizes a larger dataset of synthetic PROTAC data as well as a smaller dataset of curated PROTACs. By generating the warhead, linker, and E3 ligand, the model is less confined in its PROTAC design, thus allowing for a broader exploration of highly effective PROTACs.

Two primary model architectures will be explored, an LSTM and a Transformer using flash attention. The LSTM will be developed by integrating the state-of-the-art molecular generation framework REINVENT4 [6] into the proposed pipeline. Instead, the Transformer architecture will be based on the pre-trained NovoMolGen by Chitsaz *et al.* [10] but contain custom implementations of the training procedure. The pipeline will further standardize training to gradually shift the generated PROTAC distribution to match the target distribution. To do this, modern deep learning concepts such as TL and RL will be used to guide PROTAC generation according to the DC_{50}, D_{max}, and binary activity predicted from an ensemble of surrogate models.

Therefore, the contribution of this thesis to the research field can be broken down into four parts:

1. A pipeline for data pre-processing, model training, de novo generation, and evaluation of PROTACs.

2. Investigation into the effectiveness of reinforcement learning to guide generation further for PROTAC generation under complex objectives.

3. A collection of trained LSTM-based models capable of generating novel PROTACs for previously unseen scaffolds. Out of 10,000 generated PROTACs, the model trained on the largest dataset achieved 97.9% validity and 99.8% novelty. Furthermore, 9222 out of the 10,000 sampled SMILES were valid, unique, and canonical, of which 96.5% had predicted $DC_{50} < 100$ nM, 85.6% had predicted $D_{max} > 80\%$, and 91.7% were predicted to be active degraders for the target POI according to surrogate degradation predictor models.

4. A collection of trained Transformer-based models capable of generating novel PROTACs with previously unseen scaffolds. Out of 10,000 generated PROTACs, the model trained on the largest dataset achieved 95.6% validity and 100% novelty. Furthermore, 9112 out of the 10,000 sampled SMILES were valid, unique, and canonical, of which 99.4% had predicted $DC_{50} < 100$ nM, 98.6% had predicted $D_{max} > 80\%$, and 94.9% were predicted to be active degraders for the target POI according to surrogate degradation predictor models.

2

Related Works

Recent advancements in machine learning have provided new possibilities for automating and optimizing de novo molecular generation. Versatile molecular generation frameworks such as REINVENT4 [6] provide molecular generators targeting different aspects of constrained and unconstrained de novo design. The REINVENT4 ecosystem is comprised of four modules: Libinvent [11], Link-Invent [12], Mol2Mol [13][14], and REINVENT [15][16]. Libinvent was proposed as a method for molecular completion. Given a scaffold and attachment points, the model is tasked with decorating each attachment point with additional bonds, rings, and elements. Molecular substructures generated for each attachment point are guided via RL to maximize pre-defined objectives. Primarily, Libinvent added novelty by increasing the diversity of the decorated structures compared to previous works. Link-Invent further built on the idea of completions proposed by Libinvent, framing linker design as a conditional completion problem given two fragments. Guo *et al.* introduced Link-Invent to provide diverse linker designs by training the model on data with a broader range of atom counts for the target fragments. Differences in data preprocessing also added novelty compared to other works. The Mol2Mol module was designed for molecular optimization. Given a molecule, the model altered the structural composition of a molecule whilst penalizing large deviations from the original structure. As such, molecules could be optimized for properties without significant changes to the original molecule. Finally, REINVENT is REINVENT4's framework for unconditional molecular generation through recurrent neural networks. REINVENT was pre-trained on molecules from ChEMBL and fine-tuned using RL to alter generation towards desirable properties for de novo drug design.

Several general-purpose molecular models have also been developed. In 2025, Chitsaz *et al.* [10] introduced a family of Transformer-based foundation models trained on 1.5B molecules from the ZINC dataset. Leveraging their Transformer architecture, they showed that downstream-goal orientation for several optimization targets could be improved compared to conventional alternatives. Furthermore, their foundation models accomplished state-of-the-art results, outperforming earlier frameworks as well as domain-specific generative models across both unconstrained and goal-directed generation. Additionally, Formont *et al.* [17] successfully applied reinforcement learning using group relative policy optimization (GRPO) [18] to train a 24B parameter Transformer-based LLM. By utilizing GRPO, they achieved state-

of-the-art performance in terms of top-1 score, significantly outperforming state-of-the-art models in identifying a high-scoring candidate molecule.

Recently, generative models have shown success in PROTAC-specific tasks. PROTAC-Splitter proposed by Ribes *et al.* [19] has shown success in splitting PROTACs into its separate substructures: the linker, warhead, and E3 ligand using a Transformers. Additionally, recent research has focused on developing surrogate models in order to estimate the degradation efficiency of a given PROTAC [20]. Today, such surrogate models have shown promising performance in estimating PROTAC degradation metrics such as D_{max}, DC_{50}, and IC_{50} directly from molecular and biological features. This enables rapid approximation of experimental outcomes compared to conventional methods. Specifically, TACK by Ribes *et al.* [9] introduced three model ensembles for PROTAC degradation prediction, predicting DC_{50}, D_{max}, and binary activity ($DC_{50} < 100$ nM & $D_{max} > 80\%$).

Most work in the field of PROTAC generation has focused on designing the linker conditioned on ligands. Relevant to this thesis, Zheng *et al.* [7] proposed in 2022 the PROTAC-RL framework for producing novel PROTACs by generating a linker given the SMILES (Simplified Molecular Input Line Entry System) strings of the warhead and E3 ligand as input. PROTAC-RL uses a Transformer-based model for generating linkers as a sentence completion task. Subsequently, memory-assisted reinforcement learning is applied with the aim of guiding the model to generate PROTACS with optimized pharmacokinetic (PK) properties. Moreover, Li *et al.* [21] proposed PROTAC-INVENT, a novel generative model for 3D linker design of PROTACs. Conventional methods focused on 1D and 2D aspects to assess the linker's induced potency, ignoring structural aspects of the PROTAC's complex trinary structure present in 3D. PROTAC-INVENT decoupled the linker construction into two distinct parts: linker construction and 3D binding optimization. First, a generative model was trained to generate valid linkers given an E3 ligand and a warhead. Complete PROTACs were then transformed into their 3D counterpart and compared against reference PROTACs known to have favorable docking properties to the POI and E3 ligase. As such, a similarity between the reference and the generated PROTAC was calculated based on their molecular features. Li *et al.* further proposed replacing the reference linker with the generated linker in order to calculate docking scores for the generated structure. Generation of new linkers was ultimately guided through RL, leveraging scores calculated from its 3D representations, as opposed to previous methods focusing on 1D or 2D representations. Additional publications such as DeLinker introduced by Imrie *et al.* [22] designed linker regions using graph-based generative models.

A limited number of publications have proposed methods targeting complete PROTAC generation. Nori *et al.* [8] published a paper where they introduced a graph-based generative model for generating complete and novel PROTAC-like structures. For the generated PROTACs, Nori *et al.* focused on optimizing the DC_{50} score by applying policy gradient reinforcement learning. The rewards for the reinforcement learning process were given by a boosted-tree-based surrogate model predicting DC_{50}

in PROTAC systems. In March 2026, Hu *et al.* [23] proposed a framework capable of de novo PROTAC design through fragment assembly and guided linker design. Initially, a dataset of fragments was created through a decomposition of molecules in public datasets. Said fragments were used as building blocks in the assembly of warheads and E3 ligands for targeted protein degraders. Finally, they leveraged a surrogate model trained to predict a binary activity signal (1 if $DC_{50} \leq 100$nM and $D_{max} \geq 80\%$, else 0) in order to guide generation of the linker region [24].

3

Theory

3.1 PROTACs

PROteolysis TArgeting Chimeras (PROTACs) are engineered molecules used for targeted protein degradation, *i.e.*, they induce degradation of a protein of interest (POI). PROTACs redirect and force the ubiquitin-proteasome system (UPS) to recognize and degrade chosen POIs. This is especially important for the degradation of harmful, disease-related proteins that are otherwise unrecognizable by the UPS. PROTACs are heterobifunctional molecules consisting of a linker connecting the molecule's two functional ligand binding regions [25]. The first ligand (warhead) binds to the POI, whereas the other binding site (E3 ligand) attracts and attaches to an E3 ligase. Therefore, a PROTAC can be seen as a linker molecule that brings the POI into close proximity of the E3 ligase. The close proximity of the E3 ligase and the POI is necessary in order to induce the disposal of the POI through the UPS [2]. This methodology is *targeted* protein degradation. Since PROTACs are engineered molecules, they are purposely designed to target certain POIs, thereby inducing the degradation of certain pre-determined proteins.

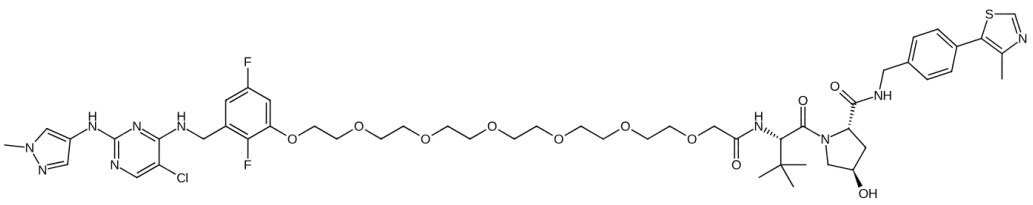


Figure 3.1: An example representation of a PROTAC. The presented PROTAC targets the JAK1 and JAK2 POIs, using the VHL E3 ligase [26].

3.1.1 PROTACs For Targeted Protein Degradation

The UPS degrades and destroys target proteins by using proteasomes. A proteasome recognizes proteins targeted for degradation through chains of ubiquitin molecules that are attached to the POI. Therefore, ubiquitin acts as a biological indicator of protein degradation. Once the E3 ligase and the POI are connected by the PROTAC, the E3 ligase attracts E2 enzymes carrying ubiquitin. The E3 ligase then repeatedly tags the POI, forming the ubiquitin chains that are recognized by the proteasome, which in turn destroys the POI rather than temporarily deactivating it. Protein degradation by the UPS is a bodily process essential for cell survival. The UPS recruits proteasomes, large protein complexes responsible for the degradation of damaged or unwanted proteins [27]. For a proteasome to recognize and degrade a POI, the POI must be polyubiquitinated [28]. Polyubiquitination means that the POI requires repeated tagging with ubiquitin proteins to form a ubiquitin chain, which is recognized by the proteasomes. In detail, this process is carried out by a stepwise sequence of three enzymes (Fig.3.2) a): an E1 activating enzyme, an E2 conjugating enzyme, and an E3 ligase. First, a free ubiquitin is covalently linked to an E1 enzyme. The ubiquitin is then transferred from the E1 enzyme to an E2 enzyme, resulting in a Ub-E2 complex. Finally, an E3 ligase binds to the POI and brings the Ub-E2 complex within close proximity, thereby facilitating the transfer and attachment of ubiquitin to the POI. After multiple cycles of ubiquitination, a ubiquitin chain is formed on the POI, thereby the POI is polyubiquitinated. Consequently, the POI is recognized and degraded by the proteasome. PROTACs promote degradation of a POI by hijacking the UPS (Fig. 3.2 b). They bring an E3 ligase and the POI into close proximity so that the POI becomes ubiquitinated, even though it would normally be unrecognized by the UPS. Furthermore, PROTACs act catalytically: after the POI is polyubiquitinated, the PROTAC detaches while remaining active, allowing it to induce the degradation of multiple POIs.

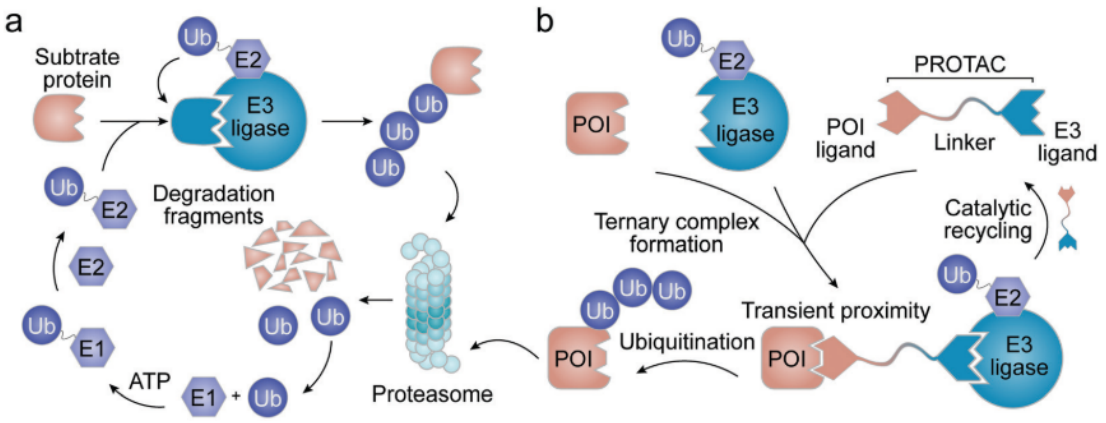
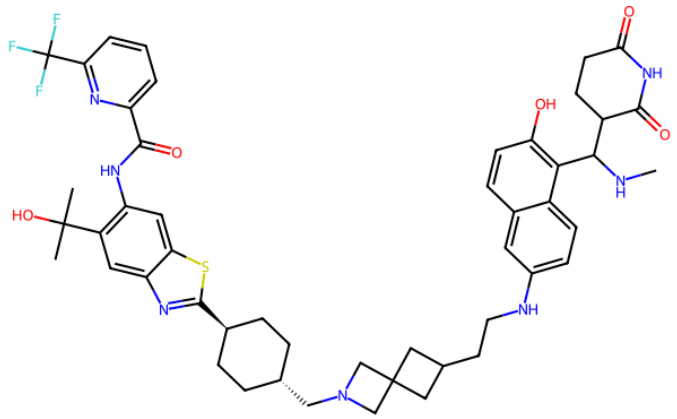


Figure 3.2: The figure visualizes the ubiquitine-proteasome system and how PROTACs are used in degrading a POI. a) Ubiquitin-proteasome system degrades arbitrary proteins in the body. b) A representation of how PROTACs induce degradation for otherwise unrecognizable POI using the ubiquitin-proteasome system [28].

3.2 Molecular Representation

The most common way of representing molecules is by using the Simplified Molecular Input Line Entry System (SMILES). SMILES was introduced by D. Weininger [29] as a way to provide machine-interpretable representations of molecules on the atom level. Each character in the SMILES sequence corresponds to either an element, bonds between atoms, or a symbol representing structural aspects of the molecule, such as rings. Figure 3.3 shows a graphical representation of the SMILES string for a molecule. There are often multiple ways to represent a molecule with SMILES. For example, ethanol can be represented as "CCO", "C-C-O", "OCC", etc. For coherence, there exist *canonicalization* algorithms that take this into account and always return one standardized SMILES representation for each molecule, referred to as *canonical* SMILES [30]. Furthermore, *isomeric* SMILES is an extension of SMILES that allows for the representation of a compound's 3D structure. Isomeric SMILES can contain information about a compound's isotopism, double bond configuration, and chirality. By default, hydrogen atoms are not required in the SMILES string; instead, they are assumed to complete the remaining available bonds.



CNC(c1c(O)ccc2cc(NCCC3CC4(C3)CN(C[C@H]3CC[C@H](c5nc6cc(C(C)(C)O)c(NC(=O)c7cccc(C(F)(F)F)n7)cc6s5)CC3)C4)ccc12)C1CCC(=O)NC1=O

Figure 3.3: A visual representation of a PROTAC and its corresponding SMILES string.

3.2.1 Fingerprints

SMILES strings are atom-based molecular representations, where each part of the molecule is explicit, such that the molecule can be fully reconstructed given its SMILES string. For larger molecules (*e.g.*, PROTACs), an exact representation can be too costly in terms of memory and computation complexity. Therefore, there exist other, lighter representations that instead *e.g.*, encode the structural or physicochemical properties of a molecule as a bit vector of a fixed size [31]. For example, encoding a molecule's structural properties would result in a bit vector with ones at the indices corresponding to substructures present in the molecule.

Such representations are ambiguous, meaning one can not reconstruct a molecule given the representation only. One of the most commonly used fingerprints, and the fingerprint used throughout this thesis, is the *Extended-Connectivity Fingerprint* (ECFP) [32], commonly referred to as the Morgan fingerprint [33]. These are radial fingerprints that, for each atom, encode its local environment up to a given radius of atoms. ECFPs are based on the Morgan algorithm developed by H. L. Morgan in 1965 [33] to solve the problem of identifying when two molecules with different atom numberings are the same (the molecular isomorphism problem). The fingerprint implementation used in this thesis is RDKit's open-source implementation of ECFP, referred to as *Morgan fingerprint* [34].

3.2.2 Tanimoto Similarity

In drug discovery, it is oftentimes useful to quantify the similarity between two compounds. Fingerprints allow for such quantification by calculating the Tanimoto similarity introduced by T. T. Tanimoto [35], shown in Equation 3.1. The Tanimoto similarity quantifies the similarity between two binary sets. It is important to keep in mind that the Tanimoto similarity between two compounds is dependent on the type of fingerprint used and the bit vector's size.

$$T(A, B) = \frac{|A \cap B|}{|A \cup B|} \tag{3.1}$$

3.2.3 Scaffolds

Complex molecules often consist of several different elements, rings, and chemical bonds. Thus, molecules that are similar can look fundamentally different based on small differences. In 1996, G. W. Bemis and M. A. Murcko published a dataset analysis of molecules to improve understanding of common features [36]. Later, their approach for data pre-processing and standardization of molecules became known as Bemis-Murcko scaffolding. Two versions of the Bemis-Murcko scaffold were introduced:

1. Excluding chemical information such as the order of bonds, the type of atom, and hybridization

2. Including chemical information such as the order of bonds, the type of atom, and hybridization

By excluding information, Bemis-Murcko divided the original 5120 compounds into 1179 different frameworks. An example of Bemis-Murcko scaffolding can be seen in Figure 3.4. The example uses the rdkit implementation; however, no details on the specifics of the scaffold procedure are provided.

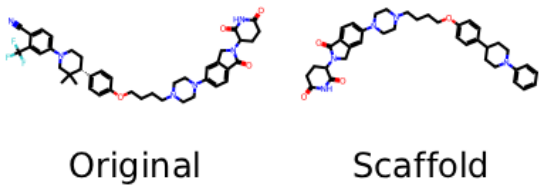


Figure 3.4: The figure shows the difference between a PROTAC and its standardized scaffold. Excess atoms and bonds are removed and only the core structure is preserved.

3.3 Chemical Properties

There are a multitude of chemical properties measurable for PROTACs and drug-like compounds in general. The properties most often highlighted are Lipinski's rule of 5 (Ro5) [37]. Properties included in Ro5 are: molecular weight (MW), number of H-bond donors (HBD), number of H-bond acceptors (HBA), and lipophilicity (LogP). Ro5 is a set of rules for what constitutes an orally absorbable drug. Although Ro5 helps assess how drug-like a molecule is, it is generally not applicable to PROTACs. This is because certain properties of PROTACs usually fall outside of the constraints given by Ro5, most prominently the molecular weight. PROTACs are long molecules; Ro5 demands a molecular weight of $\leq$ 500 Da, whereas most PROTACs have a molecular weight of $\geq$ 550 Da [38]. To help navigate *beyond* rule of 5 chemical space, Scott et al. [38] derived the *rule of oral PROTACs* from a set of active and orally absorbable PROTACs. Scott *et al.* also included additional properties such as topological surface area (tPSA) and number of rotational bonds (RotB). Furthermore, the listed physicochemical properties are computable given a SMILES string, meaning they are not required to be measured in a laboratory. This enables further evaluation of the approximate activity of AI-generated candidate PROTACs.

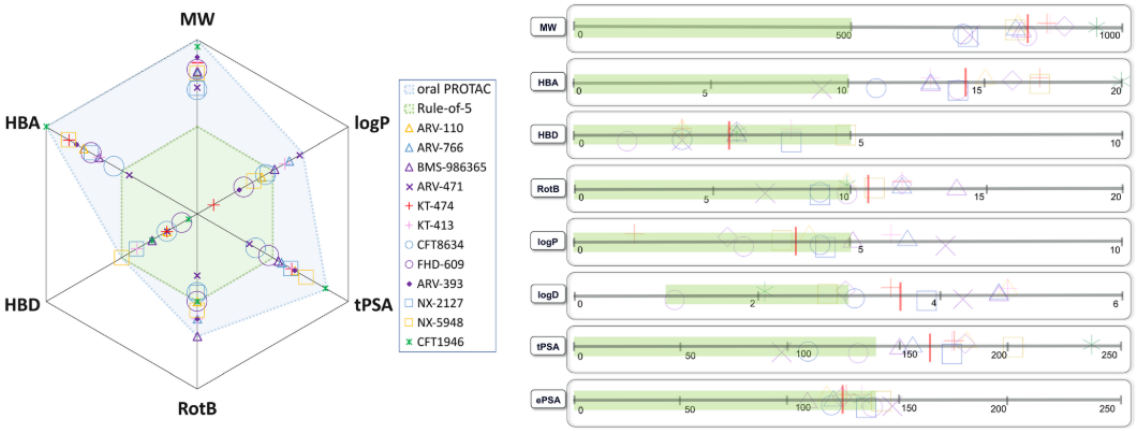


Figure 3.5: The figure shows the rule of five (marked in green) compared to the rule of oral PROTACs (marked in blue) proposed by Scott et al. [38].

Degradation is the main purpose of PROTACs; a PROTAC with suboptimal degradation performance will be considered a suboptimal PROTAC despite having other desirable properties. There are two measurements of degradation, namely and DC_{50}. D_{max} is a measurement of the degradation potential and measures the percentage by which a PROTAC can degrade a POI. For example, a D_{max} score of 100% implies that a PROTAC degraded the entire population of a POI sample [39]. Consequently, DC_{50} is a measurement of potency. Specifically, DC_{50} is a measurement of the concentration of a PROTAC required to achieve 50% of D_{max} and is measured in molar (M). Therefore, high D_{max} implies the PROTAC is capable of degrading a large amount of the POI population, whereas low DC_{50} implies high potency.

3.4 Recurrent Neural Networks

Traditional artificial neural networks are designed to transform input data into a single set of outputs through linear combinations and non-linear activation functions. Recurrent neural networks (RNNs) extend the idea of classic artificial neural networks by incorporating recurrence [40]. This permits RNNs to generate sequences of outputs $Y_{0:t} = \{y_0, ..., y_t\}$ that are conditionally dependent on the input of all previous timesteps $X_{0:t-1} = \{x_0, ..., x_{t-1}\}$ [40], [41]. The conditional generation is a key characteristic, prompting the use of RNNs in applications such as time-series analysis or continuous text generation. An overview of the architecture of an RNN is presented in Figure 3.6. The input to the network at timestep t is represented by x_t. The output of the internal state, h_t, is often calculated using several multi-layer perceptrons. Finally, y_t represents the output of the network. To facilitate computations, the first internal state h_0 does not take input or produce output. Instead, it acts as an initialization state used for the computation of the second hidden state h_1.

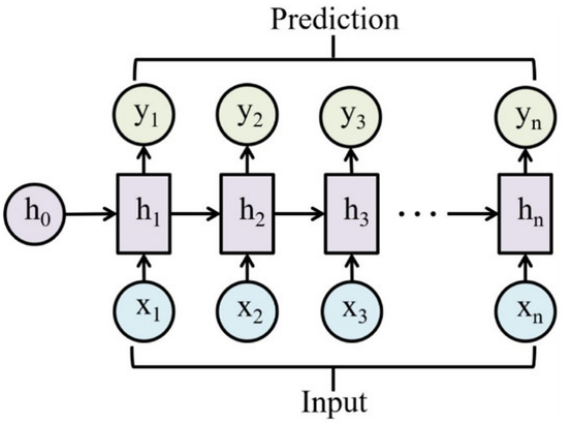


Figure 3.6: The figure shows an overview of the RNN architecture. Consisting of inputs x_t sent to the hidden state h_t and the produced output y_t at timestep t [42].

Sequential generation and pattern detection, such as text generation, requires knowledge of previous information in order to form coherent outputs. RNNs create this

coherence through their internal representation h, acting as the networks memory. At each timestep t, the previous context is encoded through a weighted combination of the new input x_t and the previous context h_{t-1}. This produces a recursive dependency where the new hidden state h_{t+1} depends on h_t and thus also all of the previous hidden states $H_{0:t-1} = \{h_0, ...h_{t-1}\}$ [40]. This enables information from previous steps to persist in memory if it aids the calculation of the next output. In addition, the activation function ϕ_h introduces non-linearity in order to improve complex learning. The complete calculation of the output from the hidden state is presented below in Equation 3.2.

$$h_t = \phi_h(w_x x_t + w_h h_{t-1} + b_h) \tag{3.2}$$

The updated context of the new hidden state h_t is further used to calculate the output y_t according to equation 3.3. By nature, the input of the next layer in an RNN is not confined to the output of the previous step. However, through sharing of its internal state, RNNs can produce coherent sequences for text generation by continuously using the output from the previous timestep y_{t-1} as new input x_t [41]. However, this implies that the network treats the output of the previous step as correct, which in some cases forces it into a hallucinating state of generation.

$$y_t = \phi_y(w_y h_t + b_y) \tag{3.3}$$

3.4.1 Gradient Propagation in RNNs

Long-term memory retention has proven to be difficult for RNNs due to their gradient calculation using backpropagation [43], [44]. During backpropagation, the gradient update for RNNs is calculated per Equations 3.4, 3.5, and 3.6, resulting in early layers having the gradient propagated through multiple activation functions before their weights are updated. This in turn gives rise to the phenomenon of *vanishing* and *exploding gradients* [40]

$$\frac{\partial \mathcal{L}}{\partial w_x} = \sum_{t=1}^{T} \frac{\partial \ell_t}{\partial \mathbf{y}_t} \cdot \frac{\partial \mathbf{y}_t}{\partial \phi_h} \cdot \mathbf{w}_y \sum_{k=1}^{t} \left(\mathbf{w}_h^\top\right)^{t-k} \cdot \mathbf{x}_k \tag{3.4}$$

$$\frac{\partial \mathcal{L}}{\partial w_h} = \sum_{t=1}^{T} \frac{\partial \ell_t}{\partial \mathbf{y}_t} \cdot \frac{\partial \mathbf{y}_t}{\partial \phi_y} \cdot \mathbf{w}_y \sum_{k=1}^{t} \left(\mathbf{w}_h^\top\right)^{t-k} \cdot \mathbf{h}_k \tag{3.5}$$

$$\frac{\partial \mathcal{L}}{\partial w_y} = \sum_{t=1}^{T} \frac{\partial \ell_t}{\partial \mathbf{y}_t} \cdot \frac{\partial \mathbf{y}_t}{\partial \phi_h} \cdot \mathbf{h}_t \tag{3.6}$$

Vanishing gradients is a recurring issue within machine learning, where the gradient vanishes as it is propagated through the network. This is caused by the chain rule used during backpropagation, resulting in the propagated value converging towards zero as the sequence grows with time. As a result of this, weights in early layers are not updated properly, resulting in less influence of early layers on the output of later layers. Therefore, the memory from early layers gradually fades away, causing recent layers to be most prominent in the calculation of the next sequence, regardless of the correlation and importance of those recent predictions. In contrast, exploding gradients occur when the value of the propagated gradient surpasses 1. Through continuous multiplications, the gradients explode to large values, causing significant shifts to the weights in early layers. Similar to vanishing gradients, this causes the memory of early layers to be applied incorrectly in the calculation of later outputs. Ultimately, both vanishing and exploding gradients harm performance. Alternative approaches to RNNs, such as long short-term memory (LSTM), have been developed to combat the long-term memory loss caused by vanishing and exploding gradients.

3.4.2 Long Short-Term Memory

The aforementioned gradient calculation greatly affects the model's ability to persist information. In order to combat memory loss, Hochreiter and Schmidhuber introduced the LSTM in 1997 [45]. The architecture of the LSTM aimed to improve gradient flow throughout the network and thus reduce problems of vanishing and exploding gradients. Compared to regular RNNs, the LSTM introduced a constant error carousel (CEC) along with gates to modulate information flow. Initially, CEC focused on how to achieve constant gradient flow for a single gate. Ultimately, Hochreiter and Schmidhuber proved that for a single, self-connected unit, this could be achieved by using an identity function $f(x) = x \ \forall x$ and a constant weight factor $w = 1.0$. Furthermore, Hochreiter and Schmidhuber extended the idea of the CEC and a single unit into multiple units by the introduction of the *input* and the *output gate*.

Introducing the input and the output gate aimed to improve the network's learning capability by combating the dual objectives of the input and output weights [45]. Initially, Hochreiter and Schmidhuber's naive approach led to conflicts during optimization. Modulating the persistence of input information through a single input weight meant that the gradient signal needed to simultaneously update the unit's ability to store necessary information and ignore unnecessary noise, causing optimization to be difficult. Similarly, using an output weight to both retrieve and disregard irrelevant information from previous units also led to suboptimal performance. Therefore, the input and output gates introduced by Hochreiter and Schmidhuber aimed to decouple the dual functionality of the original input and output weights through the introduction of multiplicative gates.

The core idea was to store information in a memory cell, c, thus constructing a CEC with constant gradient flow through the unit conditioned on the unit not receiving new input or error signals [45], [46]. This allowed the content of the memory cell to be

altered by the input and output gate. The multiplicative input gate was designed to reduce the influence of noisy inputs by blocking access to the memory cell for irrelevant unit inputs, thus providing an answer to *when* to store information. If the inputs are determined to aid in calculations, the input gate sends the information to the memory cell, which determines *what* should be stored, *i.e.*, which parts of the information are relevant. The output gate acts similarly to the input gate. It determines *when* to send information from the current unit's cell memory to the next unit. Likewise, if the information is determined to be relevant for the next unit, then its cell memory weights are responsible for determining *what* is relevant. The gates therefore control the ability to access a unit's cell memory.

More recently, A. Gers *et al.* extended the original LSTM architecture by incorporating the *forget gate* to boost performance [46]. The aforementioned CEC, input, and output gate allowed the original LSTM to preserve information indefinitely. However, due to the increasing amount of information stored in the memory cell as time progresses, the original LSTM could suffer from both blockage of input signals to the cell state, along with a gradient collapse to the original back propagation through time (BPTT). In both cases, this could lead to an underperforming network, unable to use its provided context in the most beneficial way. In order to combat this, the forget gate was introduced. The forget gate aimed to reset the cell memory of the network in the presence of outdated information, for example, across subtasks. The key change was the replacement of the constant weight factor of the CEC, $w = 1.0$, with the new forget gate. Through initialization, the forget gate was required to have a similar magnitude as the constant weight factor, $w = 1.0$, thus making the cell state behave similarly to the CEC at $t = 0$. A. Gers *et al.* defined the forget gate to be learned during training, therefore allowing for a dynamic reduction in activations in between subtasks. Ultimately, this meant that the memory cell could be reset based on training data, thus mitigating the issues of input blockage and collapse to BPTT.

A visualization of the architecture of the LSTM is presented in Figure 3.7. In the figure, c is the cell state which stores relevant information in memory, h is the internal state used in the computation of the output, and x is the input to the unit. Each gate: the forget, the input, and the output gate is represented by a multiplication symbol. The left-most multiplication symbol is the forget gate, resetting the memory before new information is sent to the memory if previous information is outdated. The second multiplication gate is the input gate, which combines the activation of the input through both a sigmoid and a tanh activation function before relevant information is added to the unit's memory. Finally, the output gate combines the sigmoid activation of the input and the hidden state, along with a tanh activation of the cell's memory. This output is used both in the calculation of the unit's output y along with the new representation of the internal state h.

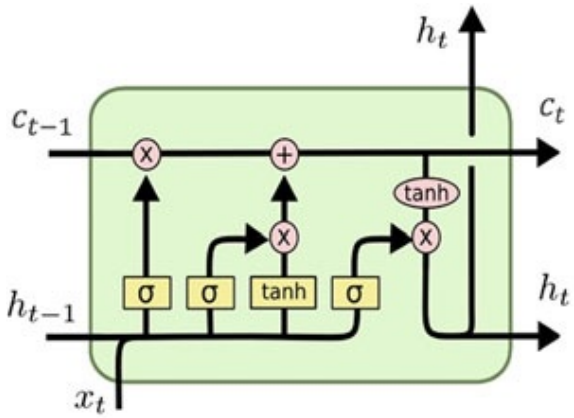


Figure 3.7: Shows the architecture of the LSTM introduced by Hochreiter and Schmidhuber [45] and further modified by the introduction of the forget gate by A. Gers *et al.* [46]. The cell, c, stores information about the context and its content is altered by the forget gate and the input gate. Finally, the output gate uses the stored information to produce a new internal state.

3.5 Transformers

The Transformer architecture for neural networks was introduced by Vaswani *et al.* as a way to tackle the difficulty of parallelizing training for conventional LSTMs and simultaneously improve long-range token dependencies [47]. At the core, Transformers rely on an attention mechanism that enables tokens of the input sequence to attend to each other, such that the meaning of a token is altered based on the tokens that surround it. For example, the semantic meaning of the sentence "... right ..." is vastly different if it is preceded by "human" instead of "turn". In order to shift the semantics of a token, the attention mechanism relies on the query (Q), key (K), and value (V) vectors to map the input into a contextualized output. The mapping is presented in Equation 3.7 as per the original equation of Vaswani *et al.* [47]. Furthermore, the key-query dot product is scaled with the key and query dimensionality d_k in order to preserve gradient size for the softmax activation.

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{3.7}$$

By constructing the attention mechanism as matrix multiplication across the key, query, and value vectors, the learned token representation can be computed in parallel, leading to the so-called multi-head attention [47]. For multi-head attention, the key, query, and value vectors are projected to h lower-dimensional heads using linear projections learned during training. Each head then constitutes a parallel attention layer such that a subspace representation can be learned more effectively.

The original Transformer architecture presented by Vaswani *et al.* features several

instances of the multi-head attention as seen in Figure 3.8, including a masked variant [47]. The Transformer is split into two distinct parts: an encoder (left in the image) and a decoder (right in the image). The encoder is responsible for adding external context to the model. In a translation setting, the encoder learns a representation of the source language sentence, whereas the decoder focuses on the target language but leverages information from the target language through the encoder. Both the encoder and the decoder use an embedding layer to transform the input tokens into a fixed internal dimensionality vector d_h. Additionally, each token is embedded using positional encoding such that the model is able to learn the tradeoff between locality bias and long-range token dependencies. The first layer in the decoder is a masked attention layer. It is used to update the semantics of the tokens generated so far. Using masked attention ensures that the semantics of token t_i cannot be adjusted based on a later token t_j if $i < j$ in the generated sequence. After learning a representation of the currently generated sequence, the decoder passes its internal representation to the next multi-head cross-attention. The cross attention combines the information from the encoder with the information of the generated sequences of the decoder. For a translation task, this makes the model learn to acknowledge the parts that are translated, the parts that are not, and how the languages relate. Finally, the decoder uses this information to construct a normalized probability vector over the vocabulary. Said probability vector is then used to determine the next token in the generated sequence according to the decoding algorithm.

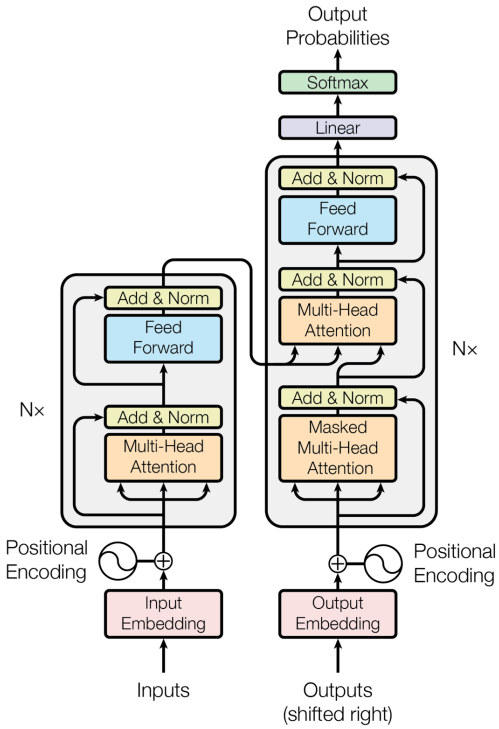


Figure 3.8: This figure shows the original Transformer architecture introduced by Vaswani *et al.* [47]. The architecture consists of two different parts: the encoder and the decoder. The encoder can be seen to the left, encoding useful information for the generation. The decoder (to the right in the image) sequentially adds its generated output to its input in order to produce coherent outputs.

3.6 Autoregressive Token Generation

Output generation through probabilistic autoregressive models, including LSTMs, has been applied in various fields, for instance, natural language processing (NLP) [48]. In NLP, a probabilistic approach to text generation decouples the direct token prediction into two distinct steps: (1) Assigning a probability distribution over the token vocabulary V and (2) Selecting the next token based on the distribution. The decoupling of probability assignment and token selection aims to provide additional control over the generation step, such that the underlying model does not have to be retrained to alter generation. Instead, during training, the model learns to assign probabilities to each token in the vocabulary based on the internal representation conditioned on the context. Thus, the output of the model is represented as a normalized probability vector $Y \in \mathbb{R}^{|V|}$ over the entire vocabulary. The token selection step ultimately decides which properties are favorable during generation by transforming the input vector through a decoding function $f : \mathbb{R}^{|V|} \to \mathbb{W}$. Several decoding algorithms have been introduced, for example *greedy decoding*. Given a probability distribution over the vocabulary, greedy decoding selects the next token greedily by maximizing each local decision. This is achieved by continuously selecting the token with the highest probability assigned by the model as presented in

equation 3.8.

$$y_t = \arg\max_y P(\ y \mid Y) \tag{3.8}$$

3.6.1 Beam Search

The idea of greedy decoding was further expanded on by the popularization of *beam search*. Beam search extends on greedy decoding by maintaining multiple candidate solutions at once, called beams, where each beam represents a unique and high-scoring candidate solution [49]. Furthermore, the candidate solutions are extended during generation such that, for each timestep t, $|V|$ number of branches are created for each of the beams. As such, each beam is extended by all unique tokens in the vocabulary, forming $|V|*|B|$ alternative solutions. In order to confine the otherwise indefinitely growing search space, only the $|B|$ highest-scoring branches are kept as candidate solutions for the next iteration. By leveraging beam search, a breadth-first search is essentially carried out in order to find high-potential sequences that are kept as starting nodes for further evaluation rather than discarding all but the highest-scoring sequence at time t, as per greedy decoding [48].

For human text generation, utilizing beam search or greedy decoding has been shown to cause issues of repetitive subsequences throughout generation [50]. Fu *et al.* showed that repetitive outputs stem from the fact that a subset of words and characters are highly influential in natural language, for example, the word "the" is frequently used as a prefix to several words in English. Thus, maximizing the likelihood of observing a token at each step can cause conditional generation to get stuck in cycles of repetitive generation.

3.6.2 Sampling

Sampling is an alternative decoding strategy aimed to increase the randomness of the generated sequence and consequently mitigate the repetition problem [51]. Given the output vector $Y \in \mathbb{R}^{|V|}$ from the model, the decoding algorithm normalizes the output vector such that each token is assigned a non-zero probability. In the widely used *temperature sampling*, a temperature coefficient is added to the normalization step in order to control the amount of randomness. The normalization step of temperature sampling can be seen in Equation 3.9. Notably, as per the equation, the higher the temperature $t \in [0, 1)$, the more uniform the probability distribution becomes. Conversely, a low temperature pushes the sampling to behave closer to greedy decoding. Given the normalized probability vector Y, the decoding function f selects the next token through sampling of the said probability vector $y \sim f(Y)$.

$$p(x = V_l | x_{1:i-1}) = \frac{\exp(u_l/t)}{\sum_{l'} \exp(u_{l'}/t)} \tag{3.9}$$

3.7 Transfer Learning

Transfer learning is a fundamental concept in machine learning aimed at utilizing knowledge obtained from a *source task* in order to generalize towards a new *target task* [52]. The idea of transfer learning for machine learning draws inspiration from psychology; learning a new topic is fundamentally easier if knowledge in a similar topic is already established [53]. In the context of statistics, transfer learning can therefore be seen as using an informed prior rather than initializing a new uninformed prior. Successful transfer learning is, however, conditioned on the domain similarity of the source and target tasks. Without sufficient overlaps in the domain, knowledge learned in the source task is not applicable to the target task. However, transfer learning on domain-similar tasks can reduce the quantity of data required in order to train a performant model. Consequently, small, high-quality datasets can be leveraged where data scarcity would otherwise hinder the training of a model for the target task.

3.8 Reinforcement Learning

Reinforcement learning is the process of learning how to achieve a goal based on experience gathered through interactions with the environment [54]. The learner is called *agent* and everything outside the agent that it interacts with is called the *environment*. The agent-environment interaction can be described as a Markov Decision Process (MDP) where the environment contains states in which the agent resides. While in a state, the agent decides to take an action, and the environment responds to said action by presenting the agent with a reward. This reward is calculated by a user-defined reward function based on how optimal the action was in terms of the goal. Along with the reward, the environment transitions the agent into a future state based on the action taken. This future state can be the same as the agent was in previously. Hence, each action can be seen as a state transition. In order to achieve the goal, the agent learns to select actions that maximize its cumulative reward.

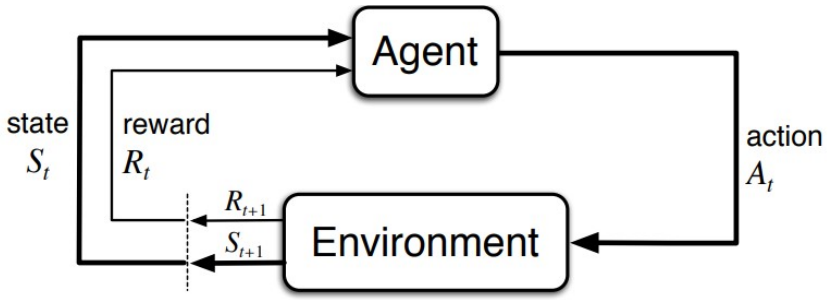


Figure 3.9: The figure shows the agent-environment interaction for reinforcement learning [54]. An agent takes continuous actions in the environment to update its state and receive rewards.

In detail, the agent and environment interact in discrete time steps $t = 1, 2, 3, ...$ In each time step, the agent is presented with a state $S_t \in \mathcal{S}$, where $\mathcal{S}$ is the

set of all available states in the environment. In state S_t, the agent chooses an action $A_t \in \mathcal{A}(S_t)$, where $\mathcal{A}(S_t)$ is the set of all available actions that the agent can take in state S_t. As a consequence of action A_t, in the following time step, the environment transitions the agent to state S_{t+1} and presents the reward R_{t+1}. Hence, a reinforcement learning episode is represented by a chain of states, actions, and rewards in the form

$$S_0, A_0, R_{t+1}, S_{t+1}, A_{t+1}, R_{t+2}, S_{t+2}, ...$$

The agent decides which action to take according to its policy π,

$$\pi_t(a|s) = Pr[A_t = a|S_t = s]. \tag{3.10}$$

Equation (3.9) denotes the probability that the agent chooses action a in state s at time step t. Hence, in reinforcement learning, the agent learns by updating its policy based on experience to maximize its cumulative reward.

3.8.1 Return

The cumulative future reward $G_t = R_{t+1} + R_{t+2} + ... + R_T$ earned from the current time step t to the final time step T is called the *return*, and often in reinforcement learning the goal is to maximize the expected return $\mathbb{E}[G_t]$, often also called the return G_t [54]. For *episodic* tasks, tasks easily broken into distinct episodes of fixed length T, this notion of return is suitable. However, for *continuous* tasks, where $T \to \infty$, this notion is problematic. Therefore, for continuous tasks, there is instead the notion of *discounted* return

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + ... = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \tag{3.11}$$

where $0 \leq \gamma \leq 1$ is the discounting factor.

3.8.2 Reward Function

The reward, also called score, received by an agent after taking an action is calculated using a reward function. This is a user-defined function designed such that it encourages actions leading towards the goal and discourages actions leading away from the goal. A reward function can be simple *e.g.,* give a reward of 1 if the agent wins a game of chess and 0 if it loses, or they can be increasingly complex, capturing more detailed information such as how optimal or suboptimal individual draws were in games of chess, or how fast the agent won a game. Hence, a reward function can be designed in many ways, which gives the user the freedom and responsibility to design a suitable one. However, this poses a major challenge in RL, namely reward hacking. Skalse *et al.* [55] formally define reward hacking as "a phenomenon

where optimizing an imperfect proxy reward function, $\mathcal{R}'$, leads to poor performance according to the true reward function, $\mathcal{R}$". Here, the true reward function, $\mathcal{R}$, is defined as the actual measure of performance that reflects the desired results *i.e.,* the reward function for learning as intended. Then, reward hacking is interpreted as when the agent learns to cheat the reward function such that it receives maximum rewards, but does not learn to solve the task at hand.

An example of this in practice is when Clark and Amodei [56] trained an agent to play a boat-racing game, where it achieved a higher final score the more targets it hit during the race. Instead of learning how to finish the game the quickest with the most targets hit, it instead found a patch of the race track where it could infinitely drive around in a circle to hit unlimited targets for an infinite reward. Reward hacking can be mitigated through careful reward function design, but still remains an open problem in reinforcement learning.

3.8.3 Value Functions

In reinforcement learning, the agent aims to maximize the return. Since the return is the cumulative reward over time, using a greedy strategy that, in every state, selects the action with the greatest immediate reward is not optimal. This since taking an action yielding a lower reward in one stage may lead to a larger cumulative reward in the future. Therefore, the agent needs to be able to estimate the expected return in order to evaluate how good a policy is in the long run. The *state* value function

$$v_\pi(s) = \mathbb{E}_\pi[G_t|S_t = s] = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \middle| S_t = s\right] \tag{3.12}$$

defines the expected return of being in state s and following policy π thereafter [54]. Similarly, the *action-value* function

$$q_\pi(s,a) = \mathbb{E}_\pi[G_t|S_t = s, A_t = a] = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \middle| S_t = s, A_t = a\right] \tag{3.13}$$

defines the expected return of being in state s, taking action a and following policy π thereafter. $v_\pi(s)$ is used by the agent to select actions, while $q_\pi(s,a)$ is used for updating the policy. These functions are learnable by the agent, and they can be estimated through different methods depending on the problem at hand. Consequently, the optimal state value- and action-value functions are defined as

$$v_*(s) = \max_\pi \; v_\pi(s) \tag{3.14}$$

$$q_*(s,a) = \max_\pi \; q_\pi(s,a). \tag{3.15}$$

A policy is optimal if and only if it is greater than or equal to all other policies. Formally, a policy π is said to be greater than another policy π' if and only if $v_\pi(s) \geq v'_\pi(s)$, $\forall s \in \mathcal{S}$. There may be more than one optimal policy π_*, and thus the agent's goal is to learn one such policy. $v_*(s)$ and $q_*(s,a)$ are shared by all optimal policies π_*, meaning that despite there being multiple optimal policies π_* there exists exactly one unique optimal state value function $v_*(s)$ and exactly one unique optimal action-value function $q_*(s,a)$.

3.8.4 Function Approximation

The previous definitions assume tabular representations of state values and action-values for each state-action pair [54]. However, in most applications the state and action spaces are often too large or complex to allow tabular representations. Then, the problem becomes generalizing to produce a good approximation over a large space with experience provided by a limited subset of the state space. This since most states encountered in large spaces have never been encountered previously. Hence, the only option is to use the experience of a subset of the state space to try to generalize to future states. Function approximation addresses this by representing these values using parameterized functions, such as linear models or neural networks. Instead of storing a separate value for each state or state-action pair, the agent learns parameters that map states or state-action pairs to predicted values. The parameterized state value function is defined as

$$\hat{v}(s, \mathbf{w}) \approx v_\pi(s) \tag{3.16}$$

where $\mathbf{w} \in \mathbb{R}^d$ is a weight vector. Similarly, the parameterized action-value function is defined as

$$\hat{q}(s, a, \mathbf{w}) \approx q_*(s, a) \tag{3.17}$$

with weight vector $\mathbf{w} \in \mathbb{R}^d$.

3.8.5 Policy Gradient Methods

Reinforcement learning methods using value functions for action selection are called *value-based* methods. Another common set of reinforcement learning methods is *policy-based* methods, which directly learn a parameterized policy to select actions without consulting a value function [54]. The parameterized policy

$$\pi(a|s, \boldsymbol{\theta}) = Pr[A_t = a | S_t = s, \boldsymbol{\theta}_t = \boldsymbol{\theta}] \tag{3.18}$$

with parameter vector $\boldsymbol{\theta} \in \mathbb{R}^d$, denotes the probability of selecting action a while being in state s at time t with parameters $\boldsymbol{\theta}$. Policy gradient methods may use value

functions in order to learn the policy parameters $\boldsymbol{\theta}$; however, value functions are not used for action selection. Policy gradient methods learn the parameters $\boldsymbol{\theta}$ based on a loss function $J(\boldsymbol{\theta})$, which is then maximized by approximation using gradient ascent:

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \alpha \nabla J(\boldsymbol{\theta}_t) \tag{3.19}$$

with step-size parameter α and $\nabla J(\boldsymbol{\theta})$ being referred to as the policy gradient. This notion introduces a concern: $J(\boldsymbol{\theta})$ depends not only on the chosen actions, but also on the state distribution, which is determined by the environment and is typically unknown. The policy gradient theorem [54]

$$\nabla J(\boldsymbol{\theta}) \propto \sum_s \mu(s) \sum_a q_\pi(s,a) \nabla \pi(a|s,\boldsymbol{\theta}) \tag{3.20}$$

provides an analytical expression of the policy gradient, not dependent on the usually unknown state distribution. This allows for the approximation of the policy gradient needed for the agent to learn the policy parameters $\boldsymbol{\theta}$. Methods that learn both a policy and a value function are called *actor-critic* methods, where the *actor* is responsible for learning the policy, whereas the *critic* learns the value function.

3.8.6 Deep Reinforcement Learning

Machine learning using neural networks with multiple hidden layers is called *deep learning*. Deep learning consists of approximating functions in high-dimensional spaces that tabular methods cannot handle [57]. Hence, neural networks are efficient at approximating high-dimensional functions. This led to deep reinforcement learning, in which reinforcement learning methods are applied to neural networks to teach them to make optimal decisions, *i.e.,* to find optimal policies, in complex, high-dimensional state spaces.

3.8.7 Reinforcement Learning in REINVENT4

REINVENT4 [6] utilizes deep reinforcement learning in order to guide its molecular generation towards molecules optimized for certain properties chosen by the user. REINVENT4's RL step works by taking as input a pretrained model as a starting point, referred to as a *prior* model. Instead of training this prior model directly, the agent in REINVENT4's RL step is a copy of this prior. Then, a policy gradient reinforcement learning approach is applied to progressively bias the molecules generated by the agent. This is because REINVENT4 uses the "Difference between Augmented and Posterior" (DAP) RL strategy [11], where the prior is kept as a baseline measurement while training the agent. This puts a constraint on how far the agent can deviate from the prior. DAP defines the loss function

$$\mathcal{L}(T) = (log\mathbf{P}_{aug}(T) - log\mathbf{P}_{agent}(T))^2 \tag{3.21}$$

where $log\mathbf{P}_{agent}(T)$ is the log-likelihood of sequence T given by the current agent. Furthermore,

$$log\mathbf{P}_{aug}(T) = log\mathbf{P}_{prior}(T) + \sigma S(T) \tag{3.22}$$

is the augmented log-likelihood, where $S(T)$ is the score (reward) of sequence T given by the reward function, and σ a scalar weight.

3.8.8 GRPO

Introduced by Shao *et al.* [18], group relative policy optimization (GRPO) is a memory-efficient version of the widely used RL algorithm proximal policy optimization (PPO). Both GRPO and PPO are used for fine-tuning LLMs, where Shao *et al.* introduced GRPO for fine-tuning LLMs for mathematical reasoning tasks. GRPO removes the need for training a separate, often equally large, value model in parallel with the policy model as in PPO. In PPO, this value model is used as a baseline in its optimization equation, whereas GRPO instead estimates this baseline as the mean reward over several sampled outputs generated for the same question. In detail (Fig. 3.10), GRPO samples a group of generated outputs $\{o_1, o_2, ..., o_G\}$ from the old policy π_{old} for a given input q. Then, the policy is optimized by maximizing the GRPO objective shown in Equation 3.23 within the sampled group. Essentially, GRPO improves the policy model by sampling a set of outputs and learning from the best samples via an internal ranking within the sampled groups.

$$\mathcal{J}_{\text{GRPO}}(\theta) = \mathbb{E}\left[\frac{1}{G}\sum_{i=1}^{G}\frac{1}{|o_i|}\sum_{t=1}^{|o_i|}\left(\min\Big(\delta_{i,t}(\theta)\hat{A}_{i,t},\ \text{clip}\big(\delta_{i,t}(\theta), 1-\varepsilon, 1+\varepsilon\big)\hat{A}_{i,t}\Big) - \beta D_{\text{KL}}(\pi_\theta \,\|\, \pi_{\text{ref}})\right)\right], \tag{3.23}$$

where

$$\delta_{i,t}(\theta) = \frac{\pi_\theta(o_{i,t} \mid q, o_{i,<t})}{\pi_{\theta_{\text{old}}}(o_{i,t} \mid q, o_{i,<t})}. \tag{3.24}$$

Here, β and ε are tunable hyperparameters, whereas $\hat{A}_{i,t}$ is the advantage calculated as the normalized reward (Equation 3.25) using the relative rewards of the outputs within each sampled group. In simple terms, the advantage of a generated output tells whether it is considered better or worse than what the model previously output for the given prompt. If $\hat{A} > 0$ for a generated output, the model updates and

learns from that prompt. Furthermore, the reference model π_{ref} is the initial and untrained policy, and the estimated KL divergence $\mathbb{D}_{KL}[\pi_\theta||\pi_{ref}]$ is subtracted as a regularization term. KL divergence is a measurement of how different two probability distributions are [58]. Hence, regularization using the KL divergence between the initial, untrained policy π_{ref} and the trained policy π_θ penalizes the training for straying too far from the reference policy.

$$\hat{A}_{i,t} = \tilde{r}_i = \frac{r_i - mean(\mathbf{r})}{std(\mathbf{r})} \tag{3.25}$$

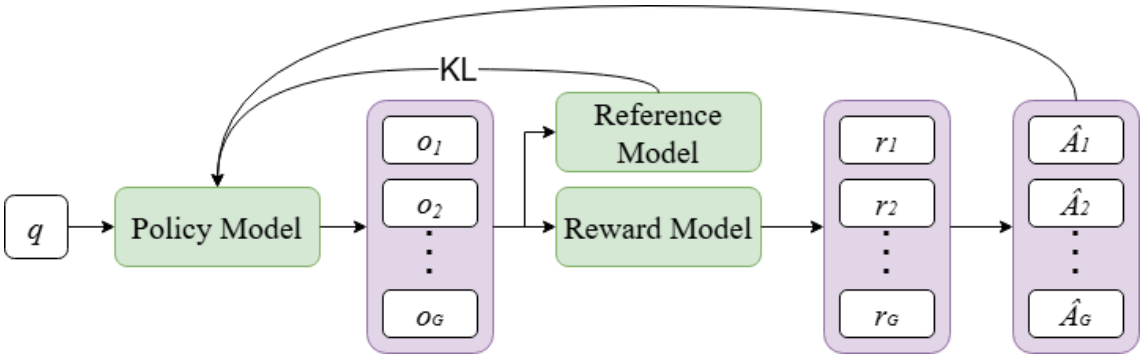


Figure 3.10: The figure shows a visualization of how the GRPO algorithm works. A policy model samples outputs which are scored by the reward model. The rewards are used to calculate the advantage of the generated outputs such that the model can be updated. Moreover, the KL-divergence for the generated sequences is calculated from the current model and a frozen reference model. Through this, a penalty can be inflicted for straying away too much from learned knowledge.

3.9 Multi-Objective Optimization

Multi-objective optimization (MOO) is the problem of optimizing for a combination of objectives at once [59]. This often requires finding the optimal tradeoffs among objectives in order to produce the optimal solution. Several strategies for MOO exist, most prominently, combining the set of objectives into a single optimization target, for example using a geometric mean or a weighted sum. The weighted geometric mean aims to combine the set of objectives into a single value by multiplying the individual objectives; this is presented in Equation 3.26 where $f_o(x)$ calculates the objective value of objective $o \in O$ for the input x. Likewise, using a weighted sum combines all objectives into an optimization target, as per Equation 3.27, by summing all objectives times their importance.

$$y = \sqrt[|O|]{\prod_{o \in O} f_o(x)^{w_o}} \tag{3.26}$$

$$y = \sum_{o \in O} w_o f_o(x) \tag{3.27}$$

3.9.1 Multi-Criteria Decision Making

The Technique for Order Preference by Similarity to Ideal Solution (TOPSIS) was first introduced by Yoon and Hwang as a post-training method for ranking candidate solutions of multi-objective optimization [60]. Each candidate solution is ranked based on its metric for a predefined set of objectives. The key idea is that, for each objective, the best candidate c^* should minimize the Euclidean distance to the best observed candidate metric. Simultaneously, c^* should maximize the distance to the candidate with the worst observed value for each objective. At first, each metric is normalized across its objective (Equation 3.28). As such, a matrix $R = \mathbb{R}^{o \times c}$ is formed, which preserves the geometric properties of each optimization target.

$$r_{c,o} = \frac{x_{c,o}}{\sqrt{\sum_{c \in C} x_{c,o}^2}} \tag{3.28}$$

Furthermore, the normalized values of each column are multiplied by an importance weight to scale ranking based on the objective's importance. Once properly scaled, the positive ideal solution A^+ is calculated. As previously mentioned, the positive ideal solution corresponds to the best values measured across candidate solutions for each objective. Therefore, it depicts the optimal solution based on the candidates. Additionally, the negative ideal solution, A^-, is calculated. TOPSIS then calculates the Euclidean distance to both the ideal and negative ideal solution for each metric and candidate. This can be seen in Equations 3.29 and 3.30.

$$S_c^+ = \sqrt{\sum_{o \in O} (r_{c,o} - A_o^+)^2} \tag{3.29}$$

$$S_c^- = \sqrt{\sum_{o \in O} (r_{c,o} - A_o^-)^2} \tag{3.30}$$

Finally, all candidate solutions are assigned a score, y_c, as per Equation 3.31. The candidate with the highest score is ranked as the best for the given multi-objective optimization.

$$y_c = \frac{S_c^-}{S_c^- + S_c^+} \tag{3.31}$$

4

Methods

4.1 Datasets

In order to train and evaluate the performance of the model, a total of three datasets were created with increasing difficulty in terms of rediscovery. In turn, each dataset consists of five splits. The five splits correspond to a held-out set of active PROTACs, a training and validation dataset for real curated PROTACs, and a training and validation dataset of synthetic PROTACs. The curated data splits were created by combining 22183 PROTACs sourced from the Targeted Protein Degradation database and two TACK datasets containing 2172 and 7703 PROTACs, respectively [9]. Furthermore, the synthetic splits were created to encourage broader exploration of the chemical space and increase the number of training samples. Synthetic PROTACs were sourced from the PROTAC-Splitter [19] project, consisting of approximately 1.3M PROTACs represented by SMILES.

The first dataset removed exact scaffold matches across the splits, ensuring that rediscovery of held-out PROTACs required rediscovery of the scaffold. The second dataset removed exact scaffold matches across the splits, along with scaffolds having a Tanimoto similarity > 0.4 compared to scaffolds in the held-out set. Finally, the third dataset applied both of the previous mitigations for data leakage, along with the isolation of the AR warhead to the held-out set.

Each dataset was initially standardized and deduplicated to reduce training bias towards specific SMILES. Standardization of the SMILES included: removing salts, neutralizing formal charges, and canonicalizing the SMILES. Afterwards, an InChIKey representing a hashed version of the InChI was calculated such that duplicate SMILES could be removed [61]. During deduplication, SMILES considered to be active were assigned a higher priority, thereby ensuring that they were kept to a higher degree. However, due to unreliable measurements of D_{max} and DC_{50} for non-TACK data, only 1054 PROTACs met the activity threshold and were assigned a higher priority during deduplication.

The held-out sets were created to provide additional metrics for evaluating the outcome of the goal-directed PROTAC optimization. As such, only active PROTACs were assigned to the held-out sets. Rediscovery or high Tanimoto similarity to a

held-out set could therefore indicate that the model is able to rediscover areas of highly active PROTACs. Initially, all PROTACs were assigned a Morgan fingerprint along with a Murcko scaffold. The radius and bit size used when calculating the Morgan fingerprint were derived from Ribes *et al.* [19]. Using a $radius = 8$ and $bit\_size = 1024$ could be seen to limit collisions between PROTACs to 0.65%, thereby accounting for the tradeoff between fingerprint uniqueness and fingerprint collisions for structurally similar PROTACS. Figure 4.1 visualizes the Tanimoto similarity to a reference molecule using the said radius and bit size. As can be seen, most PROTACs share structural similarities in the warhead and the E3 ligand, whereas differences in the linker region greatly affect the Tanimoto similarity. This is a direct result of the large radius used to calculate the Tanimoto similarity.

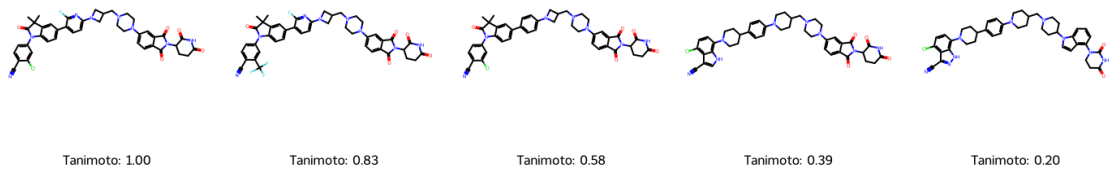


Figure 4.1: The figure shows PROTACs with different Tanimoto similarity compared to the reference (1.00 in Tanimoto similarity). As can be seen in the figure, PROTACs with less similarity mostly differ in the linker region due to the large radius and bitsize.

The first and second datasets assigned PROTACs to the held-out by first constructing a similarity matrix $S \in \mathbb{R}^{n \times n}$ of all active PROTACs. The matrix measured the Tanimoto similarity of a PROTAC's scaffold compared to the scaffold of all other active PROTACs. The said similarity matrix was then used to assign 50% of active PROTACs to the held-out sets based on their average dissimilarity to all other active PROTACs. Across all three datasets, exact scaffold matches between their respective held-out sets were removed. As such, the first dataset, "Exact Scaffold Filtered" was created. The second dataset, "Similarity-Threshold", removed additional training data where the Tanimoto similarity of the scaffold exceeded 0.4 in similarity to any scaffold in the held-out set. Thus, using a stricter requirement than Chitsaz *et al.* [10]. Training data was deliberately mixed with active PROTACs to ensure that the model could learn structural aspects of active PROTACs during TL. A total confinement of the active PROTACs would increase the difficulty of RL, as the model would have to learn structural aspects of active PROTACs from scratch.

The held-out set for the third dataset was created by isolating PROTACs with the specific AR POI. PROTACs with the AR POI were removed to increase the difficulty of the goal-oriented optimization further. Synthetic data, containing no information about the target POI, had its target POI and E3 ligase inferred. To do this, the PROTAC-splitter [19] was used to split curated PROTACs with the known target POI, E3 ligase, and cell line into their substructures: the warhead, the linker, and the E3 ligand. The synthetic data contained reliable annotations

for the substructures; thus, the PROTAC-splitter was not used on the synthetic data. Finally, the POI target and the E3 ligase were inferred for the synthetic data based on the, per substructure, maximum Tanimoto similarity to the substructure of curated PROTACs with known targets. Through this, PROTACs with the warheads targeting the AR protein could be removed from both the curated training data and the synthetic training data.

After the creation of the held-out set and the removal of scaffold-similar PROTACs, the remaining curated PROTACs were divided into a training and a validation split. An approximate 80-20 split was targeted, dividing the data based on the scaffold cluster calculated for each PROTAC. PROTACs with the same scaffold were assigned the same split. Furthermore, the synthetic splits were created to provide a broader chemical space for PROTACs, allowing for the discovery of PROTACs outside of the known curated PROTAC space. Similar to the curated datasplit, the synthetic dataset was divided into an 80-20 split. Synthetic scaffolds present in either of the curated data splits were automatically assigned to the same type of split (training or validation). This was done to prevent data leakage between training and validation when using both the synthetic and the curated training sets. Finally, the remaining synthetic data was divided based on the scaffold clusters. Table 4.1 presents the number of PROTACs in each dataset and its respective splits. Further, the figures visualize the distribution of DC_{50} and D_{max} for the training data and the held-out data across all of the datasets.

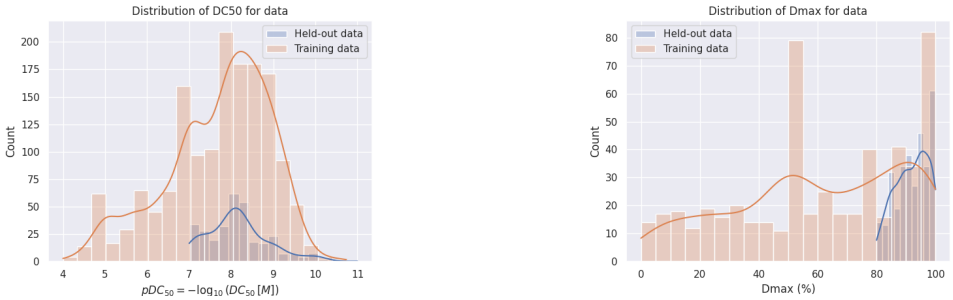
Dataset	Held-out	Curated Validation	Curated Training	Synthetic Validation	Synthetic Training
Exact Scaffold Filt.	318	4095	16788	249633	1002153
Similarity-threshold	318	3428	13174	215420	860349
No AR	358	3319	11873	173105	697827

Table 4.1: Indicated the number of samples in each split across all of the datasets. "Exact Scaffold Filt." refers to the dataset removing exact scaffold matches between each split. "Similarity-threshold" is the dataset removing PROTACs with either an exact scaffold match or a scaffold similarity to the held-out set > 0.4 based on Tanimoto similarity. "No AR" is the dataset isolating the target POI to only be contained in the held-out set.

The DC_{50} and D_{max} distribution for held-out and training data is presented in Figures 4.2, 4.3, and 4.4. For each dataset, additional information about the distribution of POI targets, E3 ligases, and cell lines for the different datasets is presented in Appendix B.1.1. Finally, a full visualization of the data pre-processing is presented in Figure 4.5 to give a more nuanced overview.

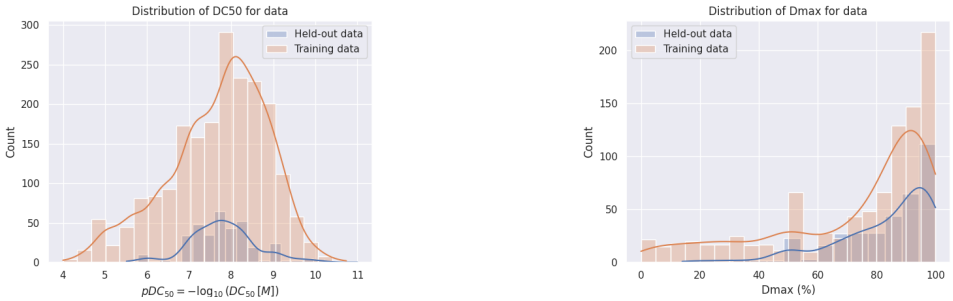
(a) **(b)**

Figure 4.2: Plot over the distribution of a) pDC_{50} and b) D_{max} for the "Exact Scaffold filtered" dataset. Neither synthetic data nor data from TPDdb is presented in the figure due to missing or unreliable measurements.



(a) **(b)**

Figure 4.3: Plot over the distribution of a) pDC_{50} and b) D_{max} for the "Similarity-Threshold" dataset. Neither synthetic data nor data from TPDdb is presented in the figure due to missing or unreliable measurements.



(a) **(b)**

Figure 4.4: Plot over the distribution of pDC_{50} and D_{max} for the "No AR" dataset. Neither synthetic data nor data from TPDdb is presented in the figure due to missing or unreliable measurements.

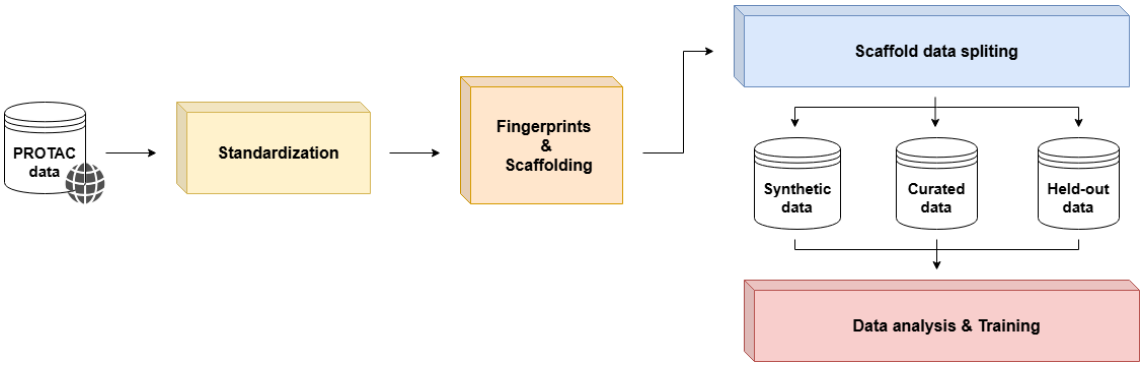


Figure 4.5: The figure shows a simplified visualization of the dataset creation pipeline. PROTAC data is first preprocessed and split based on scaffolds before it is finalized into three distinct parts: the synthetic data, the curated data, and a held-out set.

4.2 PROTAC Generation Pipeline

The thesis aim was to build a PROTAC generation pipeline and then train pre-trained prior models to generate novel PROTACs. At the core of the pipeline is AstraZeneca's general-purpose molecular generation framework REINVENT4 [6]. After data pre-processing, the PROTAC generation pipeline consists of three main parts: transfer learning, reinforcement learning, and evaluation. The transfer learning and reinforcement learning steps train a prior model to generate PROTACs optimized for user-defined properties, whereas the evaluation step evaluates the trained model against general molecular metrics as well as PROTAC-specific metrics.

Due to the recent promising results of the more modern Transformer-based architectures, the goal was to simultaneously train a Transformer-based model using the same pipeline and compare the end results to the state-of-the-art REINVENT4. Since REINVENT4 does not support training Transformer-based models for *de novo* molecular generation, a separate training pipeline was implemented for training Transformer-based models. This pipeline follows the same pattern as the REINVENT4 training pipeline. It also utilizes the exact same data processing and evaluation pipeline, as well as NLL loss for coherence. The Transformer training pipeline was implemented using HuggingFace's *Transformers* library [62] for TL and the *Transformers Reinforcement Learning* (TRL) library for RL [63].

4.2.1 Prior Models

The priors used for this thesis were both pre-trained on SMILES data, such that they already had knowledge of how to construct SMILES strings and could fully focus on learning PROTACs. The prior trained using REINVENT4 was an LSTM-based model (Fig. 4.6) with 22M parameters pre-trained on 119 million chemical compounds from PubChem [64]. The Transformer-based prior was the decoder-only model NovoMolGen_32M_SMILES_BPE model from Chitsaz *et al.* [10]. It is a

foundation model, meaning it is trained on a diverse dataset for a wide variety of purposes, and it was pre-trained on 1.5 billion ZINC-22 molecules. Furthermore, the model was built using the Llama 3 architecture [65] (Fig. 4.7), utilizing flash attention as the backbone of its 32M tunable parameters. FlashAttention is a variant of the standard self-attention mechanism that minimizes memory bottlenecks, enabling faster and more efficient training [66].

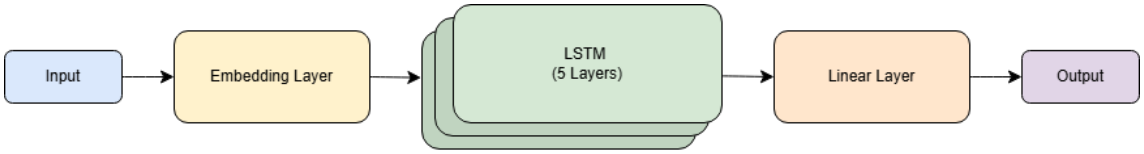


Figure 4.6: The figure shows an overview of the LSTM prior sourced from REINVENT. It consists of an embedding layer followed by five LSTM layers before an output vector is created using the linear layer.

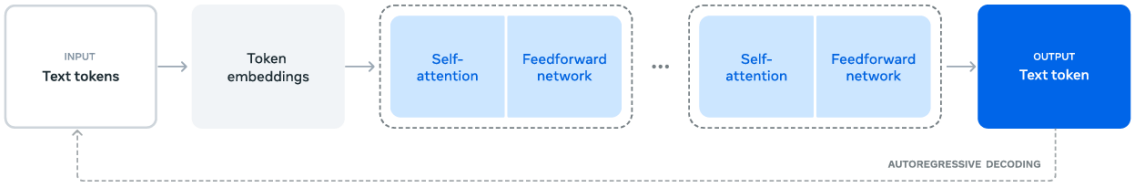


Figure 4.7: The figure shows the Llama 3 architecture, used as the backbone for the trained Transformer model [65].

4.2.2 Training

Transfer learning is used to take advantage of prior knowledge of a pre-trained model and retrain it for a different, but similar, task. The transfer learning stage in the pipeline takes a more general prior pre-trained on a wider set of chemical data. Then, the pipeline runs the appropriate training procedure according to the prior: REINVENT4's imported training pipeline for the LSTM-based prior, and the custom-built Transformer training pipeline for the Transformer-based prior. Subsequently, the prior is retrained on the pre-processed PROTAC data.

Since the prior is already familiar with the chemical space and its tokens, in the form of SMILES, it can leverage that knowledge to much more efficiently and accurately learn from the PROTAC data. Given a prior and a file of SMILES, REINVENT4's transfer learning works by computing the negative log likelihood of sampled molecules from the model against the molecules in the SMILES file [6]. The mean negative log-likelihood over all molecules is then used as the loss, which is minimized. This will train the prior to generate molecules that become increasingly similar to those in the given SMILES file. Additionally, given a validation dataset, validation loss is calculated in the same way as for training loss. Because this transfer learning approach is susceptible to overfitting, it is important to track the validation

loss to help mitigate it. The same procedure is replicated in the Transformer training pipeline. The pipeline performs transfer learning in two stages. First, the prior is trained on the large dataset of synthetic PROTACs. As mentioned, these PROTACs are fabricated PROTACs; however, they are PROTAC-like SMILES strings adhering to the same structural rules as real PROTACs. Thus, by training on this data first, the prior familiarizes itself with the PROTAC chemical space. In the second stage, this model is trained on the small dataset of curated and real PROTACs. This stage fine-tunes the model towards the space of real PROTACs.

After transfer learning, the models are subject to reinforcement learning in order to optimize their PROTAC generation towards PROTACs with optimal DC_{50} and D_{max} scores. The RL pipeline works similarly to the TL pipeline in that it recruits REINVENT4's RL pipeline for the LSTM-based prior, whereas the custom-built RL pipeline is recruited for the Transformer-based prior. As mentioned previously, REINVENT4 optimizes the model using a standard policy gradient method and then applies the DAP strategy by regularizing the RL training with the untrained prior as a baseline. This is to make sure that the trained model does not deviate too much from the original prior, causing it to lose its previous knowledge. On the other hand, the Transformer-based prior is subject to RL in the custom-built Transformer RL pipeline. This pipeline uses GRPO through the TRL library to optimize for the specified objectives. Between each stage of TL and RL, the checkpoint achieving the lowest validation loss is selected. From that, 10,000 SMILES are sampled. These SMILES are then subject to evaluation in the evaluation pipeline. For TL, this is essential for selecting the best checkpoint to start TL on the curated data. After RL, the model is fully trained, and the valuation step is therefore essential to determine the optimal final model.

4.2.3 Scoring

In order for the model to generate PROTACs with optimized DC_{50} and D_{max} properties, the scoring used in the reinforcement learning is essential. In order to score each generated PROTAC at each step, the pipeline uses the TACK [9] degradation predictor models. The TACK degradation predictor models consist of three ensemble predictors, where each ensemble consists of 25 models, and the ensemble's prediction is the average of the 25 predictions. The three ensembles predict binary activity (0-1 with score > 0.5 being active), DC_{50} (nM), and D_{max} (%) respectively. Hence, there are three separate scoring functions for scoring generated PROTACs in terms of binary activity (Equation 4.1), DC_{50} (Equation 4.2), and D_{max} (Equation 4.3), respectively. Along with the final prediction score, the ensemble predictors also quantify their own uncertainty based on the disagreement between each individual model in the ensemble. The uncertainty returned by the surrogate ensemble predictor is the standard deviation (σ) of each individual model's prediction. As such, a pessimistic and risk-averse approach is taken to the optimization. A larger standard deviation is the result of ensemble disagreement, indicating that the measurement is more unreliable. Moreover, a study by Coste *et al.* [67] showed that ensemble predictions significantly outperformed single prediction optimization. Incorporat-

ing the intra-ensemble variance was shown to both outperform pure mean-based optimization using the PPO algorithm and reduce over-optimization.

$$R_{Bin} = activity\ prediction - \sigma_{ensemble\ uncertainty} \tag{4.1}$$

$$R_{DC_{50}} = (-log_{10}(10^{-9} \cdot (DC_{50}\ prediction + \sigma_{ensemble\ uncertainty})))/10 \tag{4.2}$$

$$R_{D_{max}} = (D_{max}\ prediction - \sigma_{ensemble\ uncertainty})/100 \tag{4.3}$$

A common practice when managing DC_{50} is to convert it to pDC_{50}, which is done for the DC_{50} scoring function (Equation 4.2). Drug concentrations can differ by many orders of magnitude, hence pDC_{50} makes this easier to manage by converting to a logarithmic scale. Additionally, this makes multi-objective optimization more intuitive, as an increase in pDC_{50} means a decrease in DC_{50}. Accordingly, the ensemble uncertainty is added instead of subtracted in Equation 4.2. Consequently, to optimize for low DC_{50} and high D_{max} jointly, one can instead aim to maximize both pDC_{50} and D_{max}. Furthermore, the pDC_{50} reward and the D_{max} reward were both divided by 10 and 100, respectively, such that all three rewards are normalized to the same scale [0,1]. This prevents any reward from dominating the others.

The optimization target could be constructed in several ways for reinforcement learning. Ultimately, after empirical evaluation, using the geometric mean in combination with all reward signals was found to stabilize the training and increase model performance for the REINVENT LSTM. At first, only pDC_{50} and D_{max} were used to guide generation, but due to fluctuations in the reward signals, the binary activity was added as the primary optimization target, leading to a more stable training. Optimizing for binary activity indirectly optimizes for DC_{50} and D_{max} due to the definition of binary activity prediction. Equation 4.4 presents the optimization target as a combination of the individual reward signals. As previously mentioned, the binary activity functioned as the primary optimization target due to its stabilizing effect; thus, a weight of $w_{bin} = 0.7$ was assigned to the binary reward, and a weight of 0.15 was assigned to the DC_{50} and D_{max} reward, respectively.

$$\begin{aligned} R_{\text{total}} &= R_{\text{Bin}}^{w_{\text{Bin}}/w_{\text{tot}}} \cdot R_{\text{DC}_{50}}^{w_{\text{DC}_{50}}/w_{\text{tot}}} \cdot R_{D_{\text{max}}}^{w_{D_{\text{max}}}/w_{\text{tot}}} \\ \text{where} \quad w_{\text{tot}} &= w_{\text{Bin}} + w_{\text{DC}_{50}} + w_{D_{\text{max}}} \end{aligned} \tag{4.4}$$

For the Transformer, all three degradation scores were used to construct the total score in a similar manner. The key difference is that GRPO uses a "sum then normalize" approach where all rewards are summed to a single scalar and then normalized when calculating the advantage (Equation 3.25).

During initial training runs, the Transformer underperformed in terms of validity, uniqueness, and diversity after RL. Hence, in order to mitigate this, additional scoring functions were added for the Transformer. There were three scoring functions added in total for rewarding validity (Equation 4.5), uniqueness (Equation 4.6), and diversity (Equation 4.7). Validity is rewarded as giving a score of 1 if the generated SMILES is valid and -1 if it is invalid. Similarly, uniqueness is rewarded as giving a score of 1 if within the current group of outputs in GRPO the current SMILES is previously unseen, and -1 otherwise *i.e.,* if the SMILES is a duplicate. Furthermore, the diversity is rewarded as $1 -$ the mean pairwise Tanimoto similarity (Equation 4.7) within the current group of GRPO outputs, calculated using a radius of 8 and a bit size of 1024. The mean pairwise Tanimoto similarity is a standard way of calculating the internal structural diversity of the models generated structures [68]. These scoring functions are optimized through maximization, in the same way as the degradation reward functions. The weights used for these scoring functions were all set to 1 to be equal in worth, whereas the weights for the degradation scores were set higher to indicate higher importance. The degradation weights were set as follows: $w_{\text{Bin}} = 2.7$ and $w_{\text{DC}_{50}}, w_{D_{\max}} = 2.15$.

$$R_{validity} = \begin{cases} 1 & , \text{ if SMILES is valid} \\ -1 & , \text{ otherwise} \end{cases} \tag{4.5}$$

$$R_{uniqueness} = \begin{cases} 1 & , \text{ if SMILES is unseen within the group of generated outputs} \\ -1 & , \text{ otherwise (duplicate)} \end{cases} \tag{4.6}$$

$$R_{diversity} = 1 - \frac{1}{\#\text{pairs}} \sum_{i \neq j} Tanimoto(m_i, m_j). \tag{4.7}$$

Similarly, for the REINVENT LSTM, it was noticed during early training that it was underperforming in terms of diversity and uniqueness during RL. Hence, REINVENT's "Identical Murcko Scaffolding" [6] penalty was added as a regularization factor to penalize the model for not generating diverse enough compounds. Given a bucket size, it creates buckets that each hold a unique Murcko scaffold. Then, given a minimum score threshold, if the generated SMILES gets a score of above that, its scaffold gets stored in the corresponding bucket. Finally, if a bucket is filled above the bucket size, the following generated SMILES with the same scaffold all get a score of zero.

4.2.4 Pipeline Visualization

Here, an overview of the full training pipeline is presented, where initially a pretrained prior is trained towards the domain of PROTACs via TL. Then, the best synthetic model is trained further on curated PROTACs to mimic the real structure

of PROTACs. After TL, RL is used to improve the performance of the model further based on the predicted properties of the surrogate model. Finally, a set of PROTACs is sampled from the model, and the model's performance as well as the quality of the PROTACs are evaluated in the evaluation pipeline.

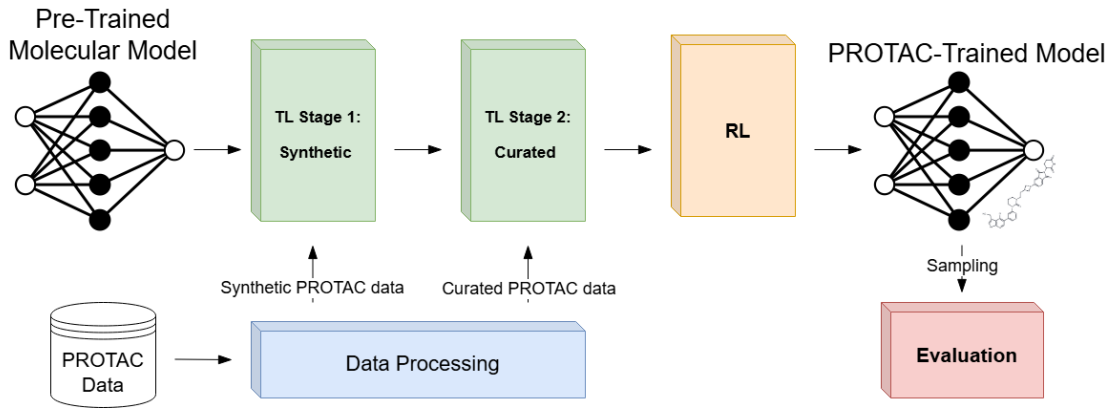


Figure 4.8: Visualization of the pipeline for complete PROTAC generation. Two pretrained models are trained with two stages of transfer learning. The models are then tuned using reinforcement learning before PROTACs are sampled and evaluated.

4.3 Evaluation Pipeline

The evaluation pipeline aims to evaluate the model's performance in terms of both performance metrics and the quality of its PROTAC generation. The model's PROTAC generation is evaluated using standard metrics for molecular generation models as well as PROTAC-specific metrics. After full training, a sampling step is run where 10,000 SMILES are sampled from the final model. These SMILES are then the basis of evaluation. First, the model is evaluated according to the standard metrics validity, uniqueness, and novelty (V.U.N.) [68]. Validity is the fraction of chemically valid SMILES the model has generated in a run, *i.e.*,

$$\frac{\#\text{valid SMILES}}{\#\text{generated SMILES}}. \tag{4.8}$$

Furthermore, uniqueness is the fraction of unique and valid SMILES generated, *i.e.*,

$$\frac{\#\text{unique} \wedge \text{valid SMILES}}{\#\text{valid SMILES}}, \tag{4.9}$$

whereas novelty denotes the fraction of unique and valid SMILES not seen before by the model, *i.e.*,

$$\frac{\#\text{ unique valid SMILES} \notin \text{training set}}{\#\text{ unique valid SMILES}}. \tag{4.10}$$

V.U.N. is a standard framework for generic molecular generation evaluation and therefore does not assess PROTAC-generation capabilities. Consequently, rediscovery is used, where a held-out set of known active PROTACs is hidden from the model. Rediscovery is then measured by counting how many of these held-out PROTACs the model can discover by itself. This not only shows whether the model is capable of generating PROTAC-like molecules, but also whether it can generate active PROTACs. Rediscovery rate is then the fraction of SMILES in the held-out set that the model can rediscover. To further evaluate the quality of the model, the evaluation pipeline also measures the internal structural diversity of the model's generated structures. This is necessary since it is possible to achieve good V.U.N. scores by minor changes in the generated molecules. Diversity can be measured in different ways, and here the diversity is calculated as the average pairwise Tanimoto distance using Morgan fingerprints with radius 8 and bit-size 1024:

$$1 - \frac{1}{\#\text{pairs}} \sum_{i \neq j} Tanimoto(m_i, m_j), \tag{4.11}$$

where m_i is the Morgan fingerprint for molecule i.

Similarly, *sample efficiency* is a performance metric for surrogate-guided RL, measured as the number of unique oracle calls needed in order for the model to reach a certain performance threshold. A high sample efficiency is therefore preferred as less computation is needed for fine-tuning the model. In this case, the sample efficiency is measured as the number of oracle calls required to reach an average binary prediction of ≥ 0.5, meaning a shift from PROTACs predicted inactive to PROTACs predicted active.

The model is also evaluated by the quality of the structures it generates. To start, the *synthesizability* of each generated SMILES is scored using the synthetic accessibility score (SAScore) [69]. SAScore is a predictive measurement of how easy a molecule would be to synthesize in a lab, given within the interval 1-10. A lower SAScore indicates easier synthesis. Furthermore, the surrogate models [9] are called to predict D_{max}, DC_{50}, and binary activity on the sampled SMILES, after which the SMILES predicted active are saved in a separate file as candidate PROTACs. Here, the activity threshold is defined as $D_{max} \geq 80\%$ and $DC_{50} \leq 100$ nM. Additionally, another separate file is generated that stores all SMILES with predicted D_{max} and DC_{50} values within these thresholds, as disagreement between these files can serve as a sanity check. Finally, the physicochemical properties of the candidate PROTACs are plotted against the rule of 5 and rule of PROTACs as in Figure 3.5.

4.4 Model Optimization and Selection

A vital part in optimizing model performance was the usage of Optuna. Optuna is a framework that automates hyperparameter selection to maximize model performance. For transfer learning, Optuna was used to optimize the learning rate, the batch size, the decay factor of the learning rate, and the frequency of the learning rate decay. Appendix B.1.1 shows performance metrics for synthetic transfer learning across 50 configurations derived by the Optuna search algorithm. During TL, TOPSIS was used to determine the best-performing model across the configurations. Using TOPSIS as opposed to the geometric mean allows for models with low scores for a single metric but very good scores for other metrics to be more competitive in the ranking. A weighted sum has also been proven to be a special case of TOPSIS [60]; as such, no information is lost compared to the weighted sum when ranking the models. The primary goal of transfer learning was to gain knowledge of the structural aspects of PROTACs, including high validity, novelty, uniqueness, and diversity. Therefore, the objectives used for TOPSIS focused mainly on these structural aspects. However, an active choice was to include the median DC_{50}, D_{max}, and activity to account for future goal-orientation. Unlike TL, where the best model was selected using TOPSIS, the final model after RL was determined entirely based on the average reward for the final epoch. The active decision not to use TOPSIS for RL was motivated by the use case of the final model: to generate the best possible PROTACs. Although validity, uniqueness, and diversity are important aspects, large sample sizes could still produce a large set of valid, unique, and diverse PROTACs. Thus, actively sampling from good regions was considered to be more significant.

5

Results

The following results section is divided into two distinct parts: (1) pipeline robustness and (2) presentation of the best-performing PROTAC generation models. First, Section 5.1 presents results for pipeline robustness by analyzing the inter-run variability for identical setups. Section 5.2 highlights the best-performing model for each training stage and backbone model, along with the results of the intra-run variability. The results presented are based on the most frequent combination of target POI, E3 ligase, and cell line in the held-out sets. This combination is presented in table 5.1. Targeting Androgen has been shown to be successful in the treatment of prostate cancer [70].

POI	POI Uniprot	E3 Ligase	E3 Uniprot	Cell Line
Androgen (AR)	P10275	Cereblon (CRBN)	Q96SW2	LNCaP

Table 5.1: Highlights the protein target, E3 ligase target, and cell line. Additionally, the POI uniprot and E3 uniprot are presented for reproducibility purposes.

5.1 Pipeline Robustness

The robustness of the pipeline was defined based on the reproducibility of results given the same configuration (hyperparameters, training step, dataset, and backbone model). Initially, an optimal set of hyperparameters was selected for each combination of configuration through the use of Optuna trials. Once the optimal set of hyperparameters had been determined for the configuration, four additional experiments with an identical setup were conducted. As such, the robustness of the pipeline could be evaluated through cross-evaluation on the 50,000 PROTACs generated between the five models. Analysis and evaluation of the robustness is based on differences in means across the different runs. This ensures that the evaluation carried out is representative of the difference between runs rather than the total variability of the system as a whole. For example, each model is evaluated based on the molecular weight of generated PROTACs. The presented mean is therefore the mean of each individual run's mean for molecular weight. Furthermore, the standard deviation between the runs is presented to give further insights into the stability. A

lower standard deviation across the metrics indicates that the model might relearn to sample from the same PROTAC distribution, reducing the likelihood of producing outlier models.

5.1.1 Chemical Evaluation Metrics

Computer-based frameworks for molecular generation are frequently evaluated using the standardized evaluation metrics presented in Section 4.3. This includes metrics such as validity, uniqueness, novelty, diversity, and the synthesizability of generated PROTACs. Said metrics are presented in figure 5.1 for the LSTM model and figure 5.2 for the Transformer model. Each metric presents the mean and standard deviation for the combination of the dataset and training stage. The different datasets are presented using different symbols (circle, cross, and triangle), and the training stages are color-coded (blue, orange, and green). Furthermore, the preferred direction for each metric is represented by the direction of the arrow.

The performance of the LSTM model can be seen to be consistent across identical runs for all datasets during the two TL stages. This indicates that the model is able to provide reliable performance for the standard metrics regardless of the randomness introduced during training. However, whilst the RL improves the validity, novelty, and synthesizability of the generated PROTACs, it exhibits less inter-run stability along with worse performance for both uniqueness and diversity. A similar trend can be seen for the Transformer in figure 5.2, where the RL appears to be less robust compared to the TL. However, for both validity and novelty, inter-run variability is also seen during TL. A more detailed presentation of these evaluation metrics is showcased in table B.1. There, exact measurements are presented for both mean and standard deviation.

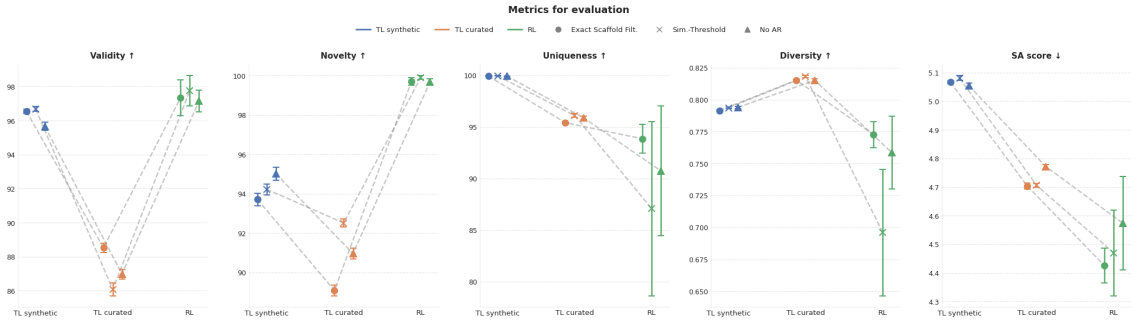


Figure 5.1: Presents evaluation metrics calculated based on the average and standard deviation of identical runs with the optimal hyperparameters for the LSTM model.

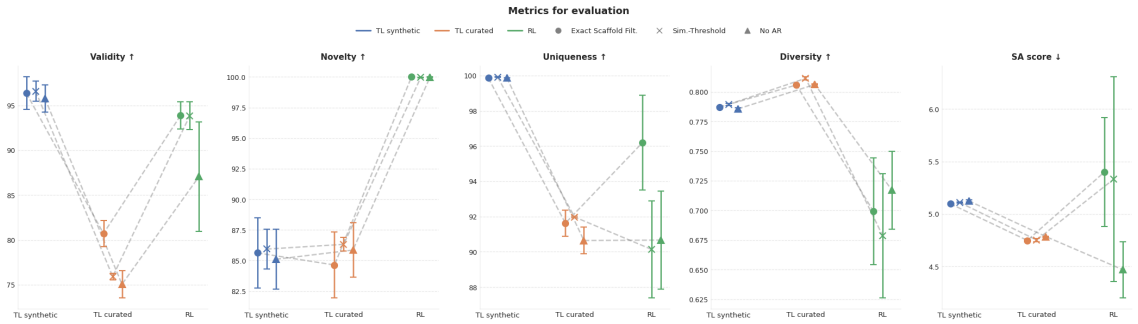


Figure 5.2: Presents evaluation metrics calculated based on the average and standard deviation of identical runs with the optimal hyperparameters for the Transformer model.

5.1.2 Sample Efficiency

The primary goal for the pipeline is to generate novel and potent PROTACs. As such, the robustness of the pipeline is also dependent on the reproducibility of PROTAC potency. Applications of RL through a surrogate model for molecular design often present the sample efficiency of a model. This refers to the number of oracle calls required to achieve a predefined performance threshold. As such, the performance in terms of the DC_{50}, the D_{max}, and the binary activity reward signal is plotted against the number of oracle calls for the LSTM model in Figure 5.3 and the Transformer model in Figure 5.4. The mean and the standard deviation are calculated for each batch of 256 oracle calls across the five identical runs.

For the LSTM, there is a clear tendency for the binary activity to improve with the number of oracle calls. It is also the noisiest in terms of inter-run variance. However, there is no significant difference between the datasets in performance. Notably, all models perform well for the first oracle call, indicating that the TL is effective in increasing the performance on its own. The Transformer shows similar performance levels for the DC_{50} and D_{max} reward across all datasets. However, the binary activity reward does not converge to the same value for all datasets. Instead, the exact scaffold filtered dataset appears to both converge to a lower value and exhibit greater inter-run variance compared to the others.

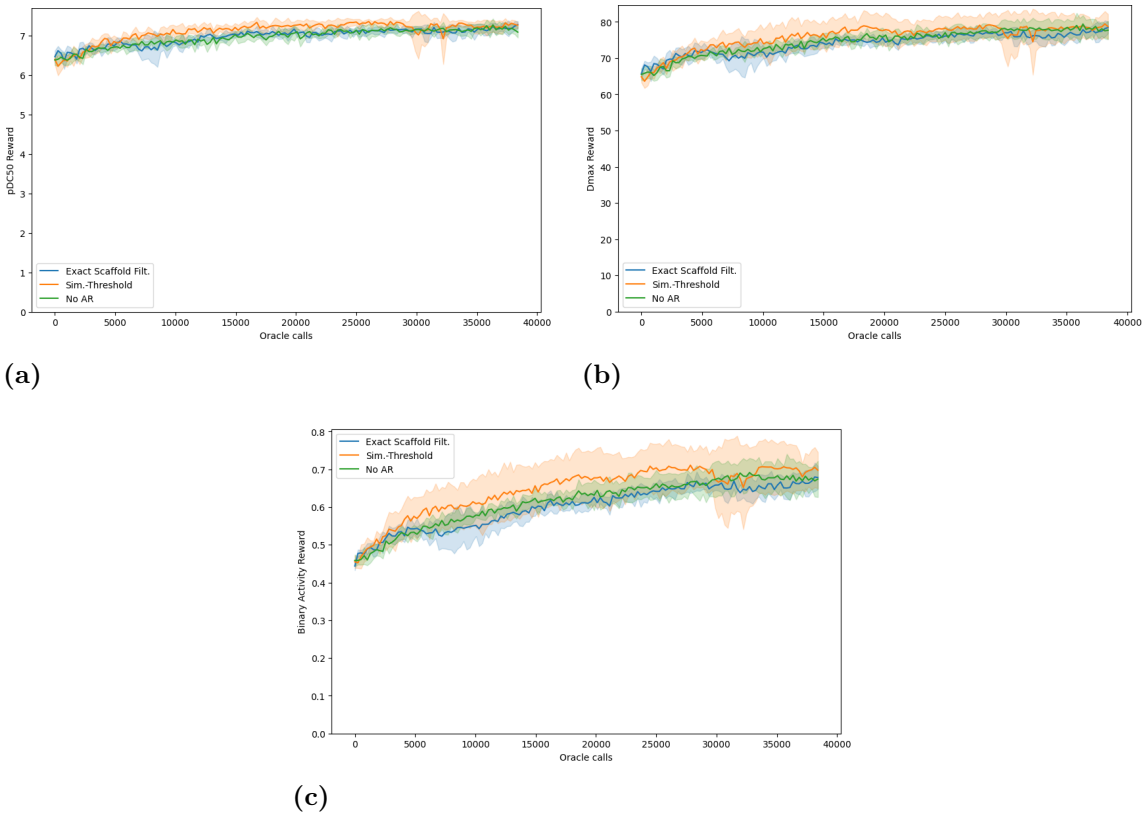


Figure 5.3: The REINVENT LSTM model's sample efficiency as the batch-average pDC_{50} a), D_{max} b), and binary activity c) reward of the generated smiles during RL. The plotted reward is the reward received by the models from the reward functions during RL training. The plot shows the mean ± 1 standard deviation for the batch-wise means across 5 identical runs.

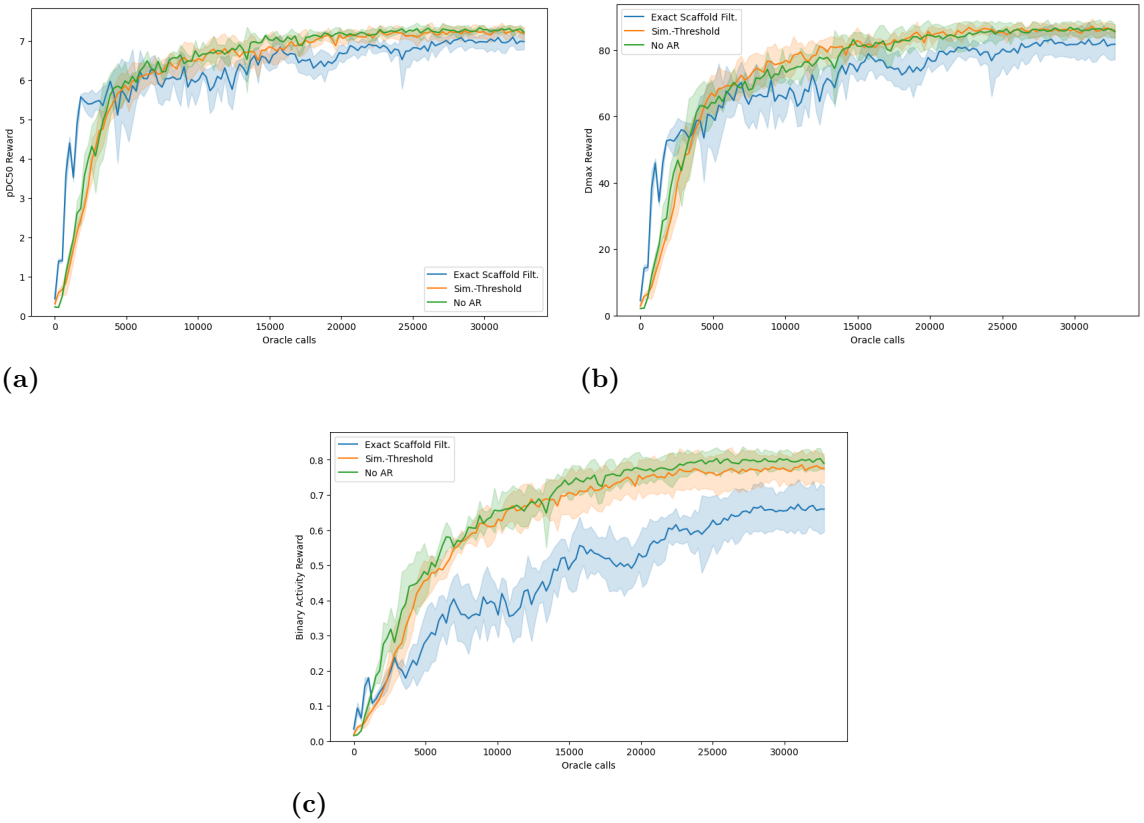


Figure 5.4: The Transformer model's sample efficiency as the batch-average pDC_{50} a), D_{max} b), and binary activity c) of the generated smiles during RL. The plotted reward is the reward received by the models from the reward functions during RL training. The plot shows the mean ± 1 standard deviation for the batch-wise means across 5 identical runs.

5.1.3 Similarity to Held-Out PROTACs

Rediscovery of held-out PROTACs is a difficult task. Instead, Tanimoto similarity can provide insights into the similarity without measuring exact matches in molecular structure. Like the sample efficiency presented in Section 5.1.2, the Tanimoto similarity presented in Figures 5.5 and 5.6 is based on the inter-run similarity to the held-out per batch of oracle calls. Moreover, the similarity is calculated as follows. For each sample within a batch, the similarity to the held-out set is measured as the maximum Tanimoto similarity to any PROTAC in the held-out set. Each run then calculates the batch average similarity based on the previously described maximum similarity.

From Figure 5.5, there is a slight improvement in the Tanimoto similarity over time. This indicates that the model is gradually shifting towards regions where PROTACs are known to be potent for the combination of POI, E3, and cell line. However, as mentioned, the difference is small and the inter-run variance is large for the LSTM. Additionally, the dataset used in TL does not seem to affect the Tanimoto similarity for the LSTM. The Transformer does not exhibit any tendency to shift generation

to PROTACs with similarities to the held-out sets. Rather, all datasets appear to converge at a similarity lower than or equal to their starting similarity.

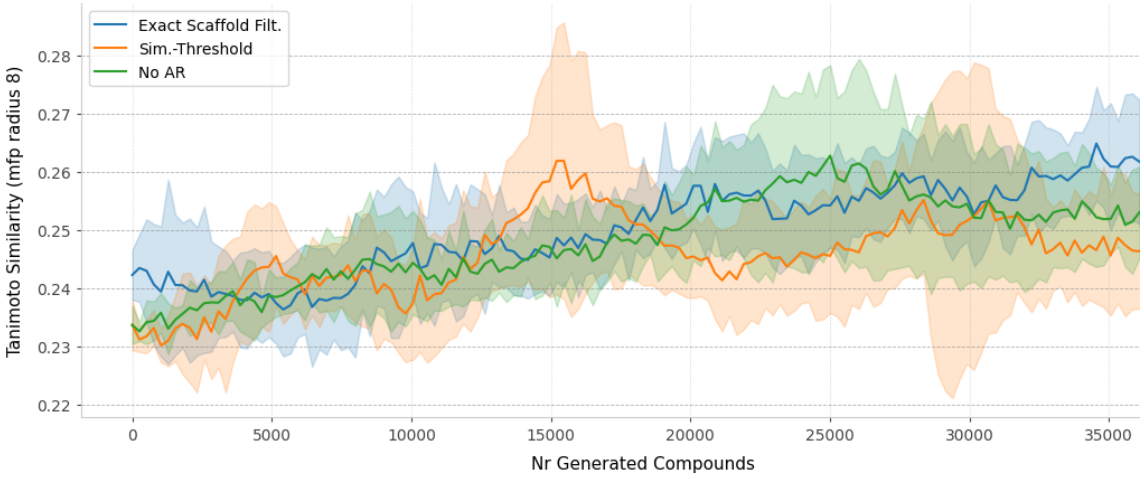


Figure 5.5: Batch average similarity to the held-out dataset for the 5 identical REINVENT runs. The plot shows, for each dataset, the mean of the means of the 5 identical runs ± 1 standard deviation.

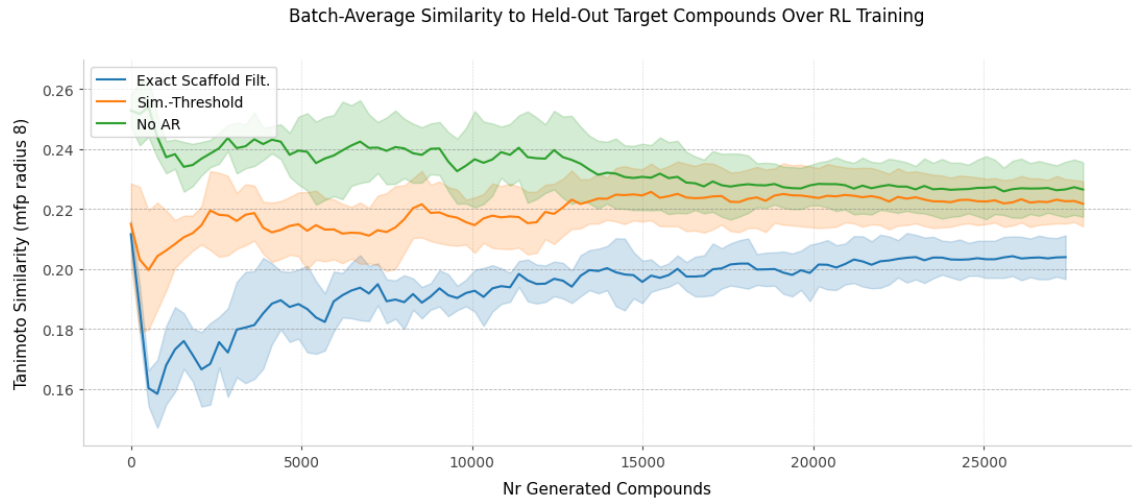


Figure 5.6: Batch average similarity to the held-out dataset for the 5 identical Transformer runs. The plot shows, for each dataset, the mean of the means of the 5 identical runs ± 1 standard deviation.

5.1.4 Physicochemical Properties

During RL, the models were not tasked with optimizing PROTACs for physicochemical properties. However, the properties can indicate if a molecule is PROTAC-like. Figures 5.7 and 5.8 present error bars representing the mean molecular weight (MW), hydrogen bond acceptors (HBA), hydrogen bond donors (HBD), rotatable bonds (RotB), topological polar surface area (tPSA), and lipophilicity (LogP). As can be

seen for both models, inter-run variance is, as expected, low for all datasets across TL. Conversely, RL has high inter-run variance for most metrics across the LSTM and the Transformer. A more detailed overview of the inter-run variability for the physicochemical properties is presented in Table B.2. A notable difference between the LSTM and the Transformer is that the LSTM's mean lies closer to its mean for curated TL across all metrics. This indicates that the LSTM deviates less from the structural aspects it has learned from real PROTACs. Additionally, it is clear that the synthetic data is captured similarly for both the LSTM and the Transformer. However, the synthetic data is slightly different than real curated data, leading to higher values across all properties for the synthetic TL step.

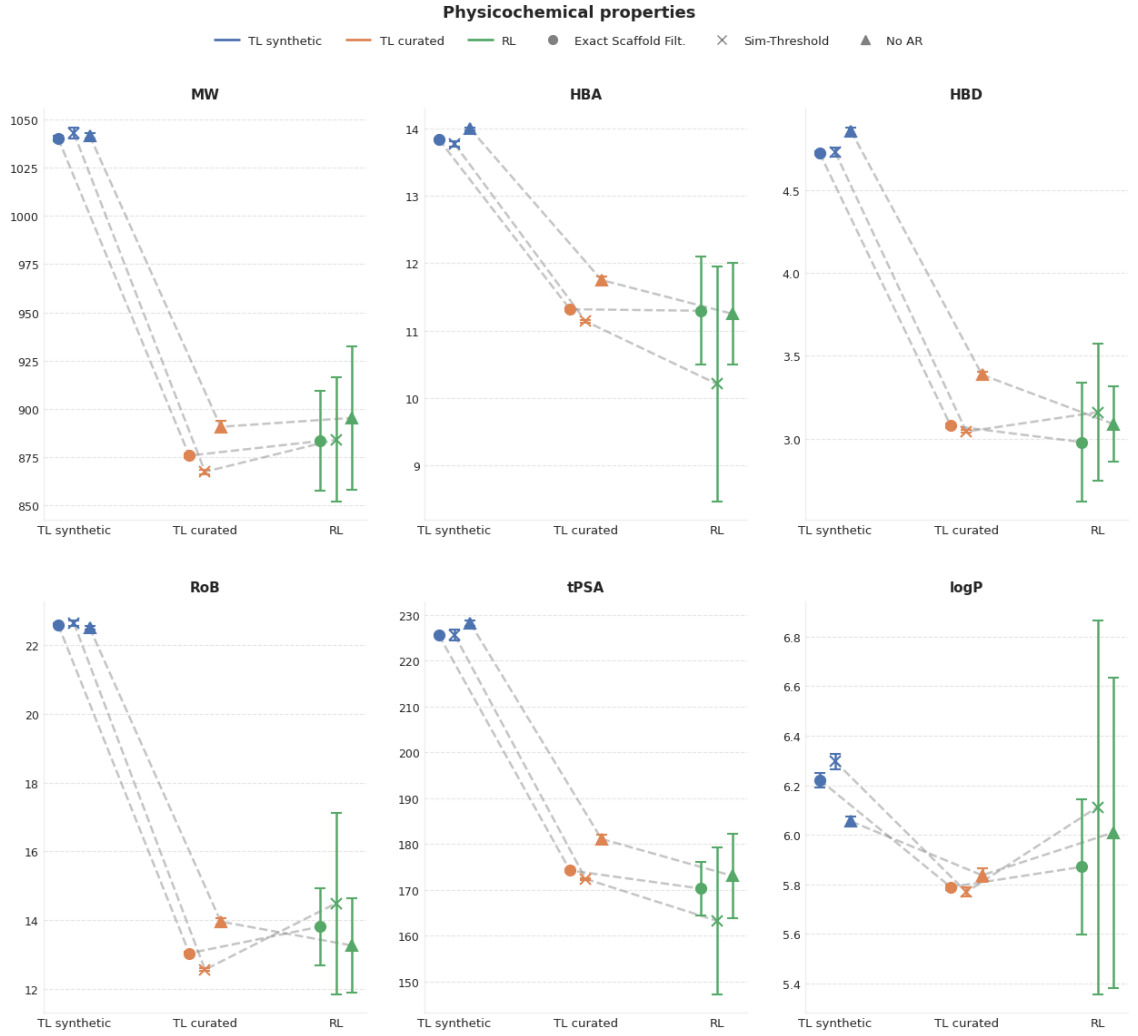


Figure 5.7: The figure shows an error-bar plot over the physicochemical properties of PROTACs generated by the LSTM model. This includes the molecular weight (MW), hydrogen bond acceptors (HBA), hydrogen bond donors (HBD), rotatable bonds (RotB), topological polar surface area (tPSA), and lipophilicity (LogP). The mean and standard deviation are calculated based on the inter-run variability.

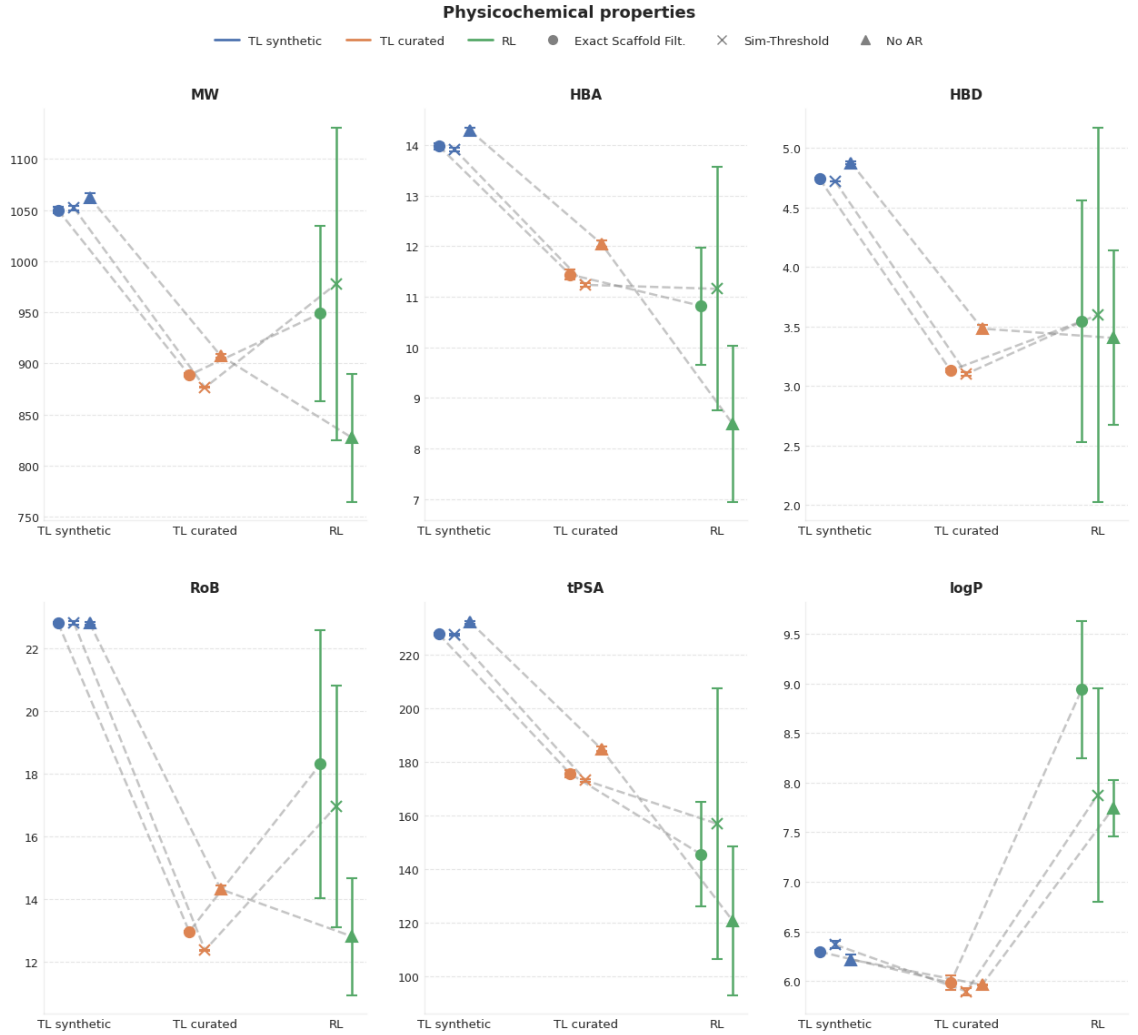


Figure 5.8: The figure shows error-bar plots over the physicochemical properties of PROTACs generated by the Transformer model. This includes the molecular weight (MW), hydrogen bond acceptors (HBA), hydrogen bond donors (HBD), rotatable bonds (RotB), topological polar surface area (tPSA), and lipophilicity (LogP). The mean and standard deviation is calculated based on the inter-run variability.

5.2 Evaluation of Best Runs

Pipeline robustness is an important aspect in the reproducibility of results. However, only a single model can be used to sample structures. Presenting the single best performance for each part ensures connection to the real world, as evaluation of PROTACs in a wet-lab is limited due to the expense. Therefore, the following section presents the results of the single best-performing model for each configuration of model, dataset, and training step. The mean and standard deviation presented in the figures correspond to the intra-run variability. As previously mentioned, 10,000 PROTACs are sampled after the completion of each training step. In order to calculate the intra-model variability, the sampled PROTACs are divided into 10

bins of 1,000 PROTACs each. The variability can thus be calculated to give an indication of how much the model performance differs when generating PROTACs in smaller batches.

5.2.1 Chemical Evaluation Metrics

Standard metrics used in academia for molecular generation are presented in Figure 5.9 for the LSTM and Figure 5.10 for the Transformer model. As such, the model performance can be evaluated against other state-of-the-art implementations in terms of validity, novelty, uniqueness, diversity, and synthesizability. As expected, the presented results share the same overall tendency presented in Figures 5.1 and 5.2 for the inter-run variability. Both TL and RL appear stable across all metrics. Together, this indicates that the best-performing models are capable of producing PROTACs with consistent performance, regardless of sample size. The metrics presented in the figures are presented with additional detail in Table B.3.

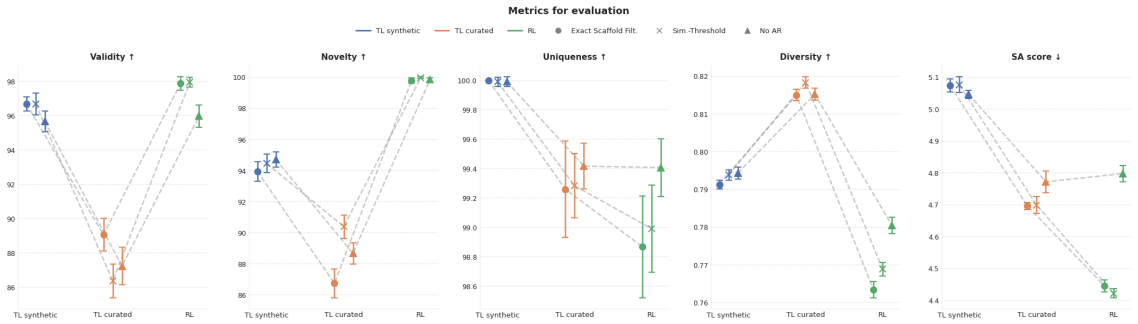


Figure 5.9: Highlights the best-performing LSTM model and its intra-model variance. The standard V.U.N metrics are presented along with the diversity and the synthesizability (SA score) of generated PROTACs.

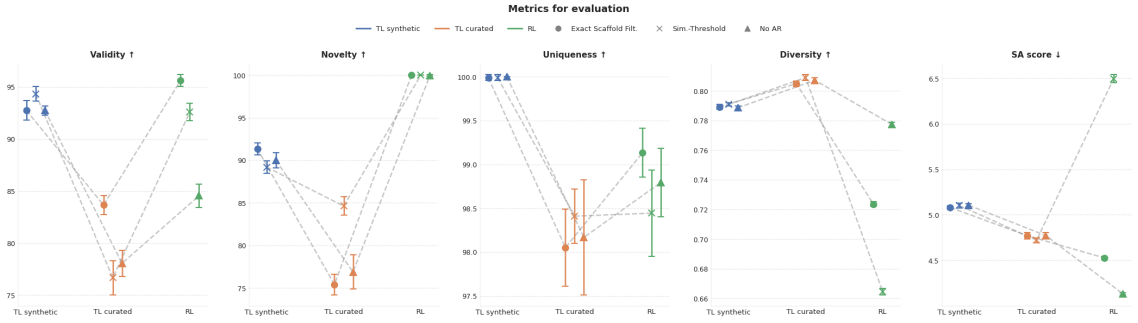
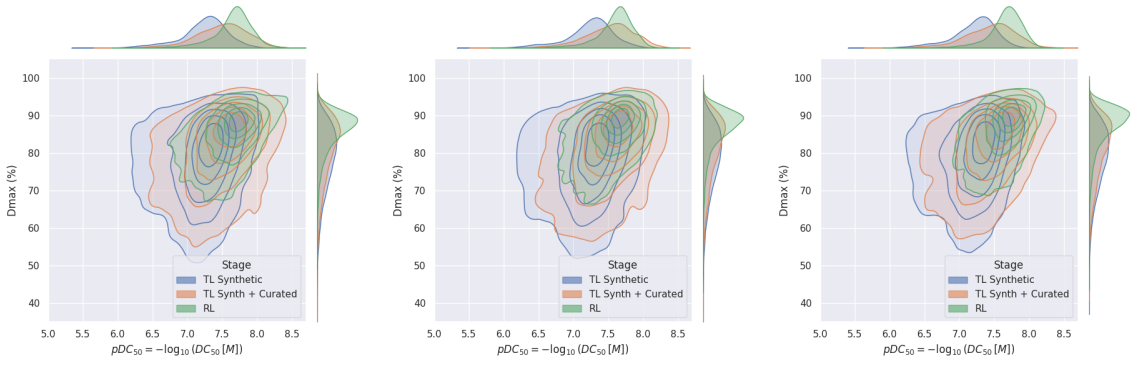


Figure 5.10: Highlights the best-performing Transformer model and its intra-model variance. The standard V.U.N metrics are presented along with the diversity and the synthesizability (SA score) of generated PROTACs.

5.2.2 Distribution of pDC_{50}, D_{max}, and Binary Activity

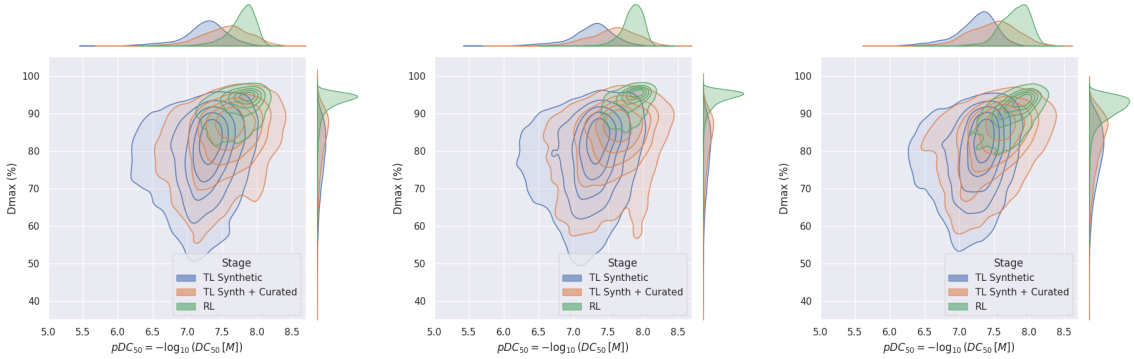
Metrics such as validity, novelty, and uniqueness are important aspects in PROTAC generation. However, the primary goal of a PROTAC is efficient degradation of the target POI. Therefore, this section presents an evaluation of D_{max}, DC_{50}, and binary activity. SMILES sampled after training are evaluated based only on the valid and unique SMILES, as those are the interesting PROTACs for real-world testing. As such, the number of SMILES evaluated is reduced from the 10,000 sampled to the numbers presented in table 5.2.

The degradation distributions for the REINVENT LSTM and the Transformer are presented in Figures 5.11 and 5.12, respectively. This allows for a direct comparison of how the generated distribution changes based on the training stage and dataset. The figure consists of a 2D kernel density plot capturing the relationship between pDC_{50} and D_{max}. An area of dense contour lines indicates a higher tendency to sample PROTACs with the specified pDC_{50} and D_{max}. To give additional insights, each axis presents a 1D kernel density plot over the specified property of the axis. As such, it becomes easier to understand how the pDC_{50} and D_{max} are distributed individually. Most notably, the distribution of generated PROTACs gets increasingly confined to a localized area throughout TL and RL. This means that the model is able to find a subset of the chemical space where it is certain about the performance of the generated PROTACs. Furthermore, there is a tendency for the D_{max} to increase throughout training and stabilize around 90% for the LSTM. The transformer achieves slightly better performance in terms of D_{max} with scores concentrated around 95%. As seen in the figure, pDC_{50} is also improved, converging around 7.75. A notable difference between the LSTM and the Transformer is the density of the generated PROTACs. The Transformer is able to confine the generation of PROTACs to an area of consistent pDC_{50} and D_{max} performance. However, since the sampling space is much more concentrated, it may not be equally diverse, as evidenced by Figures 5.9 and 5.10.



(a) **(b)** **(c)**

Figure 5.11: Distribution over pDC_{50} and D_{max} for generated PROTACs using the REINVENT LSTM across the exact scaffold filtered (a), similarity-threshold (b), and no AR (c) datasets.



(a) **(b)** **(c)**

Figure 5.12: Distribution over pDC_{50} and D_{max} for generated PROTACs using the Transformer across the exact scaffold filtered (a), similarity-threshold (b), and no AR (c) datasets.

Information about the number of PROTACs fulfilling different criteria is presented in Table 5.2. The column DC_{50} (%) represents the percentage of PROTACs generated with $DC_{50} < 100$nM. Similarly, D_{max} (%) presents the percentage of PROTACs generated with $D_{max} > 80\%$. The binary column represents the percentage of PROTACs with a predicted binary activity > 0.5, meaning the PROTAC is voted as majority active. Finally, the count column shows how many out of the 10,000 sampled PROTACs were left after standardization and removal of duplicates, from which the percentages are measured. As seen in the table, both the LSTM and the Transformer maintain the performance across all datasets. However, the Transformer can be seen to generate fewer valid and unique PROTACs when the difficulty of the dataset increases.

Model	**Dataset**	**DC_{50} (%)**	**D_{max} (%)**	**Binary pred. (%)**	Count
REINVENT LSTM	Exact Scaffold Filt.	96.46	85.55	91.47	9222
	Sim.-Threshold	96.14	88.14	89.98	9217
	No AR	97.49	88.96	91.95	9255
Transformer	Exact Scaffold Filt.	99.4	98.6	94.9	9112
	Sim.-Threshold	99.8	99.4	98.6	8550
	No AR	99.6	97.5	95.2	7755

Table 5.2: The table shows the percentage of PROTACs generated meeting the criteria for DC_{50}, D_{max}, and binary activity. The percentage is calculated based on the valid and canonicalized unique PROTACs generated for each model. Additionally, the count of valid and canonicalized unique is presented.

Across all datasets, there is a tradeoff between high pDC_{50} and high D_{max} for the sampled PROTACs. Figure 5.13 shows this tradeoff by visualizing PROTACs that are weakly Pareto optimal. A weak Pareto optimal point refers to a point that is not dominated by any other point across all metrics [71]. In the case of PROTAC generation and Figure 5.13, a weak Pareto optimal PROTAC is therefore defined as a PROTAC that is not dominated by any other PROTAC for both pDC_{50} and D_{max}.

As seen in the figure, the maximum values for D_{max} lie in the range [96, 98] for both the LSTM and the Transformer. Interestingly, the best LSTM model trained on the exact scaffold filtered dataset manages to increase pDC_{50} without sacrificing D_{max} to a greater extent than the other datasets. Moreover, the Transformer model can be seen to produce a more uniform Pareto front across the different datasets, indicating that the tradeoff is consistent regardless of the previous training procedure.

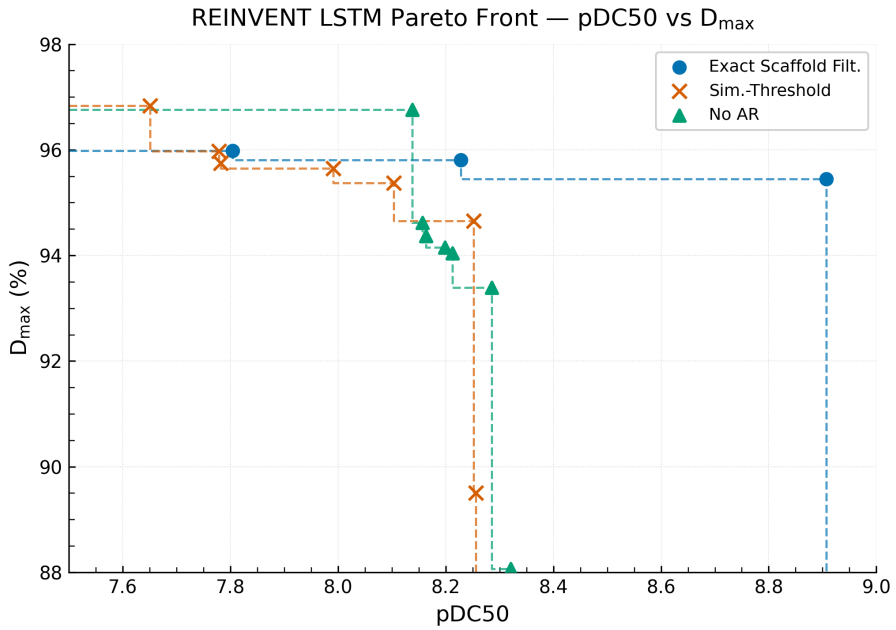


Figure 5.13: The figure shows the Pareto front plot for pDC_{50} and D_{max} for the REINVENT LSTM backbone model. The compounds that constitute the Pareto front can be found in Figures B.7, B.8, and B.9.

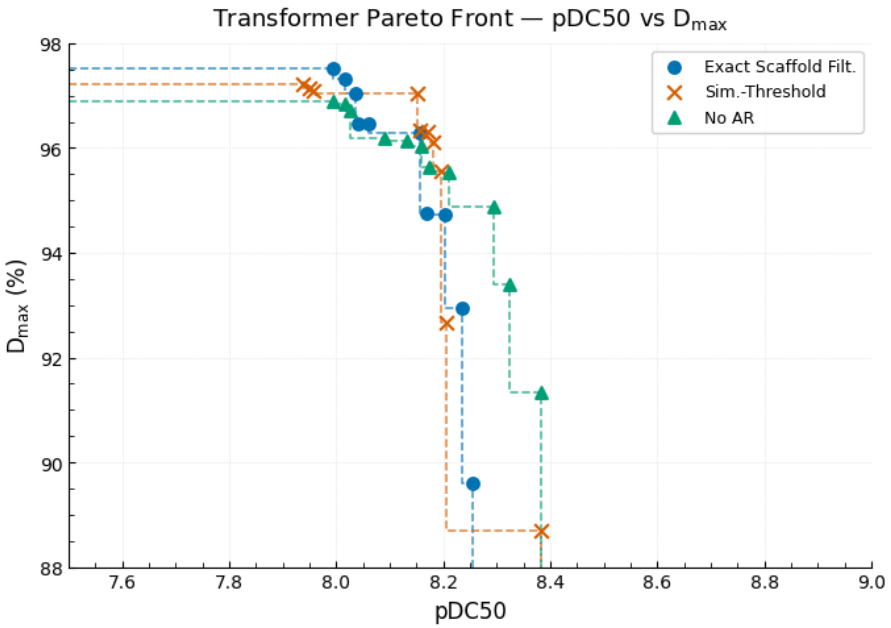


Figure 5.14: The figure shows the Pareto front plot for pDC_{50} and D_{max} for the Transformer backbone model. The compounds that constitute the Pareto front can be found in Figures B.10, B.11, and B.12.

In Figures 5.15 and 5.16, the sample efficiency of the LSTM and the Transformer models is plotted, as the batch-average reward over the number of oracle calls. Here, the surrogate reward models are treated as black-box oracles with the DC_{50}, D_{max}, and binary activity predictions of a PROTAC counted as a single oracle call to the surrogate ensemble. The plot shows that RL successfully makes the model more efficient at generating molecules that are predicted to be active. Both the LSTM and the Transformer can be seen to improve the rewards over time. The LSTM requires fewer steps in order to shift the generation of PROTACs from inactive to active according to the binary predictor. However, as seen in the figures, the LSTM starts with greater values; this is primarily due to the poor validity of the Transformer model for early stages of RL, resulting in low batch-average scores.

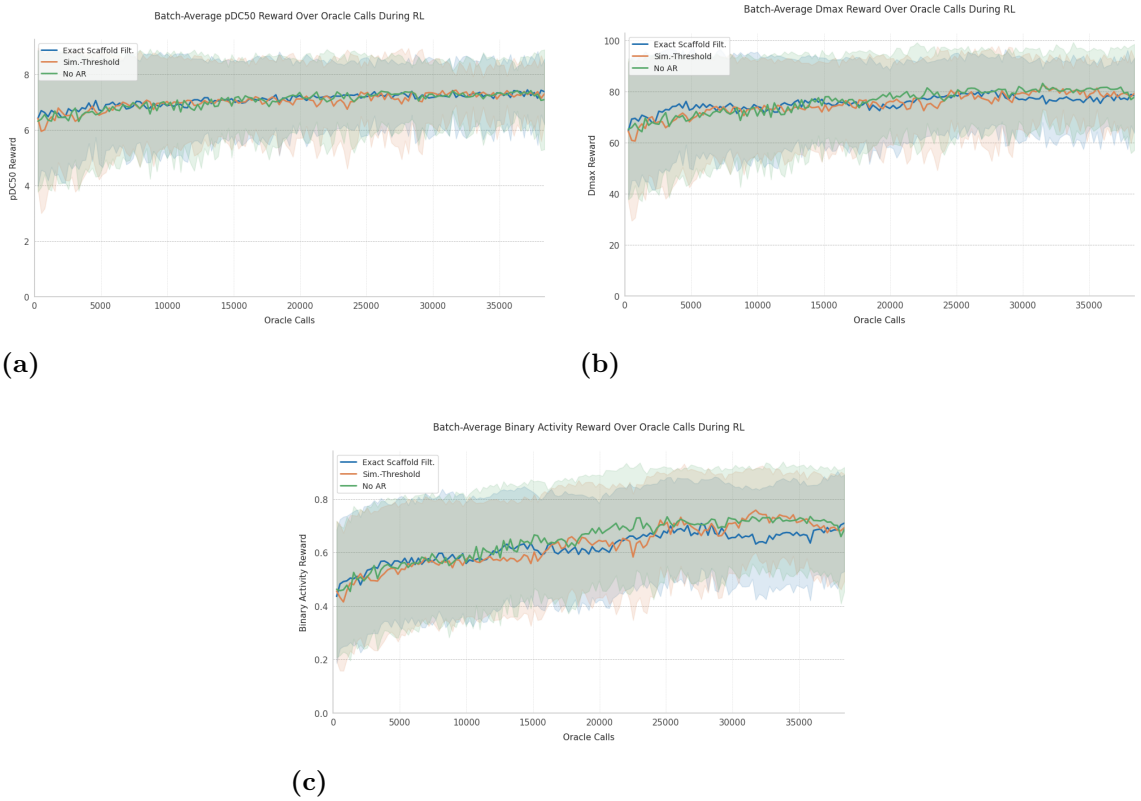


Figure 5.15: The best REINVENT LSTM model's sample efficiency as the batch-average pDC_{50} a), D_{max} b), and binary activity c) of the generated smiles during RL. The plotted reward is the reward received by the models from the reward functions during RL training. The plot shows the mean ± 1 standard deviation for the batch-wise means across 5 identical runs.

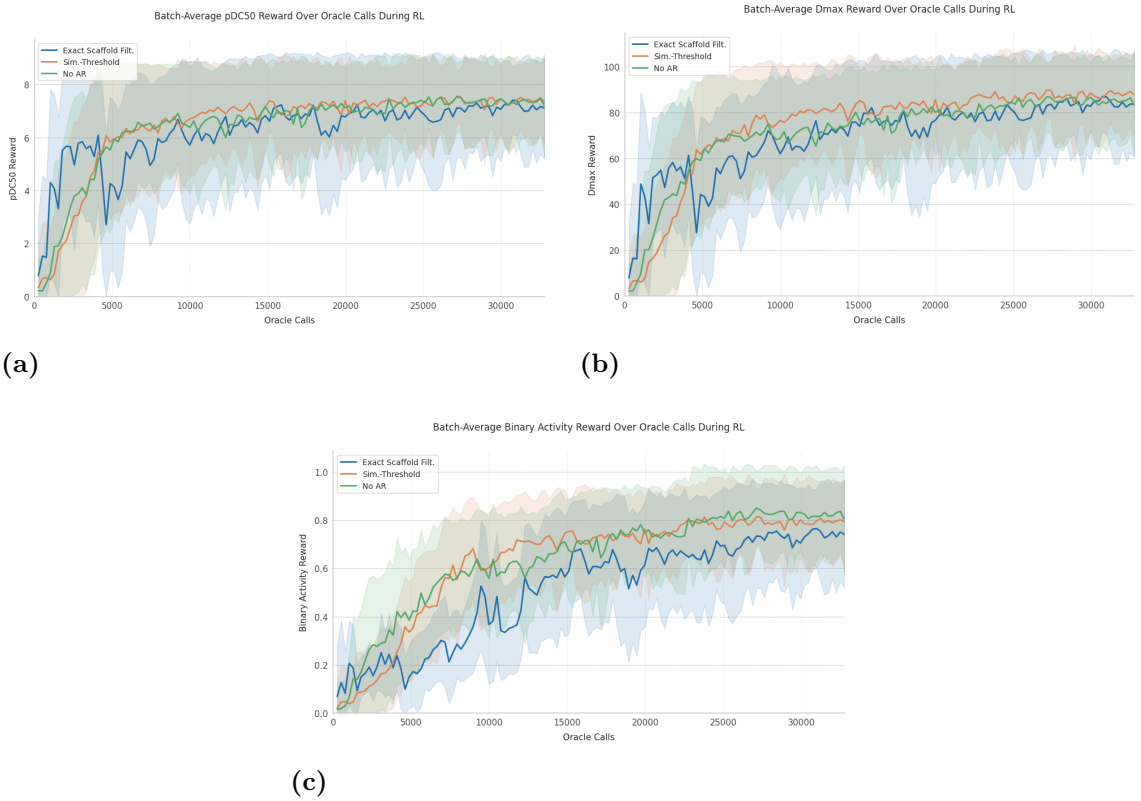


Figure 5.16: The best Transformer model's sample efficiency as the batch-average pDC_{50} a), D_{max} b), and binary activity c) of the generated smiles during RL. The plotted reward is the reward received by the models from the reward functions during RL training. The plot shows the mean ± 1 standard deviation for the batch-wise means across 5 identical runs.

5.2.3 Similarity to Held-Out set

Rediscovery of PROTACs from a held-out set is frequently used to indicate that a model is well-tuned. The key idea is that if a model can rediscover PROTACs from a set of highly active PROTACs, then it has learned to sample from a good sample space. The number of PROTACs rediscovered for each step is presented in Table 5.3. Notably, the use of RL does not result in an increased number of rediscovered PROTACs compared to the previous step, even if DC_{50}, D_{max}, and binary activity are increased throughout RL. This could be due to several factors, such as finding more optimal regions of PROTAC generation according to the surrogate model or reward hacking. Therefore, additional information about the maximum number of rediscovered PROTACs across each backbone model, training stage, and dataset is presented in Table 5.3 to give a more nuanced picture of the generation and allow for further comparison of the effectiveness of RL in boosting model performance. Each rediscovery count is presented in two parts: targeted rediscovery and total rediscovery. Targeted rediscovery refers to the number of PROTACs rediscovered with the AR POI, CRBN E3 Ligase, and the LNCaP cell line used to tune the RL. Total rediscovery instead refers to the total number of PROTACs that were

rediscovered, regardless of their target POI, E3 Ligase, and cell line. For early stages of training, the total rediscovery is a better indicator of the model's capability, as it is not trained to predict PROTACs targeting specific properties. During RL, targeted rediscovery gives a better indication of the model's ability to shift generation to the target domain. In the figure, the values are presented as *targeted rediscovery / total rediscovery*. Additionally, the increase in rediscovery between the TL steps is primarily a result of the change in data distribution. The held-out set is created from curated PROTACs; a training set consisting of curated PROTACs should lie closer to the held-out set in the chemical space than synthetic data.

	Synthetic TL			Curated TL			RL		
Backbone Model	ESF	Sim.-Thresh.	No AR	ESF	Sim.-Thresh.	No AR	ESF	Sim.-Thresh.	No AR
REINVENT LSTM	0/1	0/1	0/0	1/5	0/1	0/0	1/4	0/2	0/0
Transformer	0/1	0/0	0/0	2/7	0/1	0/0	1/5	0/0	0/0

Table 5.3: The table presents the maximum rediscovery count across training steps and datasets. The maximum count is based on the 50 Optuna trials for determining the best hyperparameters per dataset and training step. Rediscovery is defined as the number of PROTACs found within the set of 10 000 PROTACs generated after model training was completed. Additionally, only PROTACs with the sought-after target POI, E3 ligase, and cell line from Table 5.1 are counted toward the rediscovery count. ESF is short for "Exact Scaffold Filtered"

Figures 5.17 and 5.18 show the batch-average similarity of the SMILES generated by both models during RL to the subset of the held-out set with active PROTACs for the target POI, E3, and cell line combination. The plot shows that RL indeed guides the model towards generating PROTACs that are more similar to known active PROTACs for the specified target E3, POI, and cell line combination. The batch-average similarity was calculated using Tanimoto similarity with radius 8 and bit size 1024. The averaging was done for each generated SMILES in a batch, calculating its Tanimoto similarity to all PROTACS in the target subset of the held-out set. Then, the batch-average was calculated as the average of the maximum similarity score of all SMILES in the batch. Only the maximum was considered since the known active PROTACs in the target subset are somewhat diverse themselves, and a generated SMILES needs only to be similar to one of them to be considered similar to known actives.

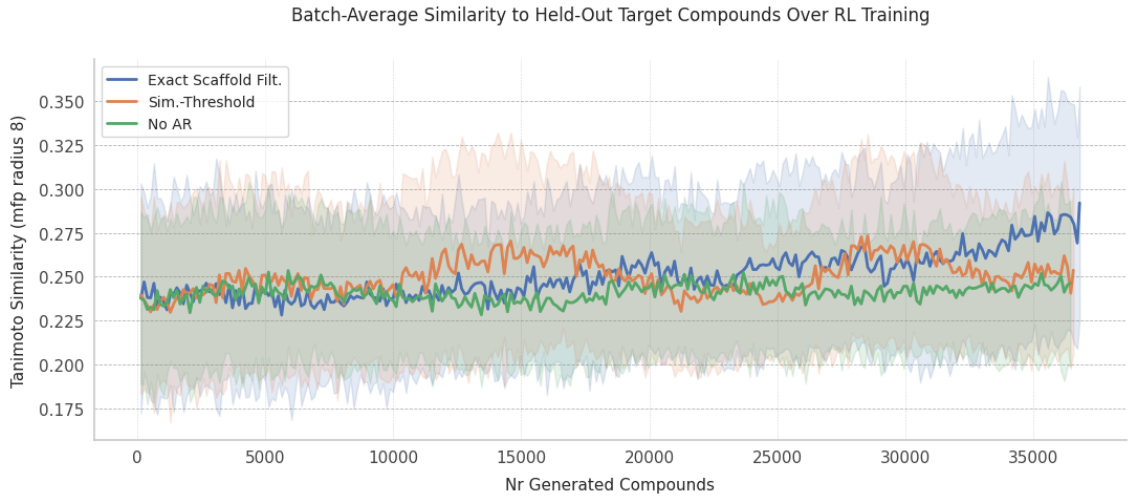


Figure 5.17: Batch-average Tanimoto similarity to the target subset of the held-out set for the generated compounds during RL for the best REINVENT LSTM model. The Tanimoto similarity was calculated using Morgan fingerprints with a radius of 8 and a bit size of 1024.

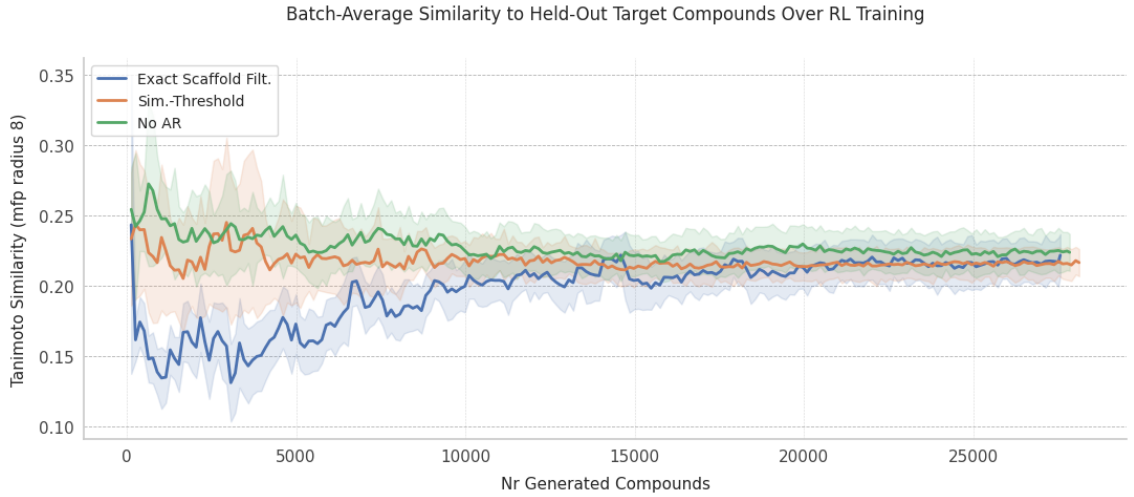
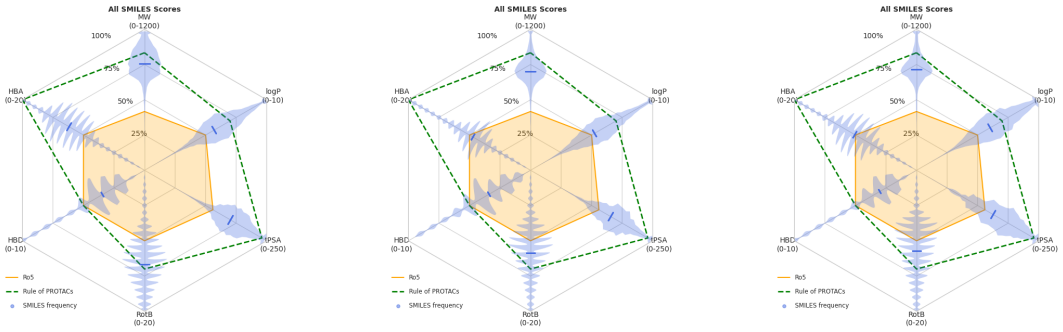


Figure 5.18: Batch-average Tanimoto similarity to the target subset of the held-out set for the generated compounds during RL for the best Transformer model. The Tanimoto similarity was calculated using Morgan fingerprints with a radius of 8 and a bit size of 1024.

5.2.4 Physicochemical Properties

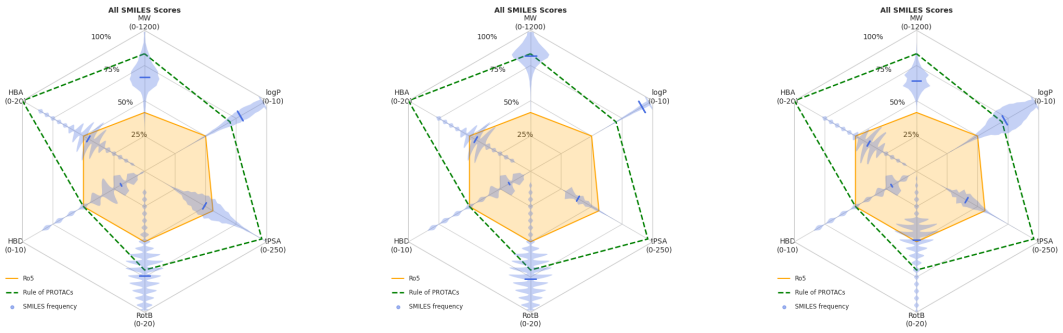
The physicochemical properties of the generated PROTACs are presented in more detail in Figures 5.19 and 5.20 to give a more nuanced overview of the property distribution. In the Figures, the generated PROTACs' physicochemical properties are compared both to the rule of five (Ro5) and the proposed rule of oral PROTACs (RoP), introduced by Scott et al. [38]. Each axis presents a violin plot over the generated PROTACs' distribution for that specific property, along with its mean. The

distribution for physicochemical properties is similar across the different datasets for the LSTM, indicating that the generated structures are similar. Additionally, the mean of all properties lies below the upper boundary presented for oral PROTACs. The distribution for the Transformer varies more across the datasets compared to the LSTM. Notably, LogP is significantly elevated, and the model trained using the similarity-threshold dataset produced heavier PROTACs after RL.



(a) **(b)** **(c)**

Figure 5.19: Summary of the physicochemical properties of the PROTACs generated using the LSTM after RL for the model trained on the exact scaffold filtered (a), similarity-threshold (b), and no AR (c) datasets.



(a) **(b)** **(c)**

Figure 5.20: Summary of the physicochemical properties of the PROTACs generated using the Transformer after RL for the model trained on the exact scaffold filtered (a), similarity-threshold (b), and no AR (c) datasets.

5.2.5 Visualization of Generated PROTACs

The figure shows the top-4 highest scoring PROTACs in terms of binary activity for the REINVENT LSTM models trained on the three different datasets. The bottom row presents the PROTACs for the target POI, E3, and cell line combination from the held-out dataset with the highest D_{max}. The coloring of the generated PROTACs indicates the degree of novelty as the similarity to any compound in the training data.

If the Tanimoto similarity (Morgan fingerprint radius 8 bit-size 1024) between the generated PROTAC and any compound in the training data is $= 1$, the generated PROTAC is not novel and is colored green. If the similarity is < 0.5 the generated PROTAC is considered novel and colored pink. Finally, if the similarity is < 1 but ≥ 0.5, the generated PROTAC is colored orange as an indication that it is novel but more similar to the training data.

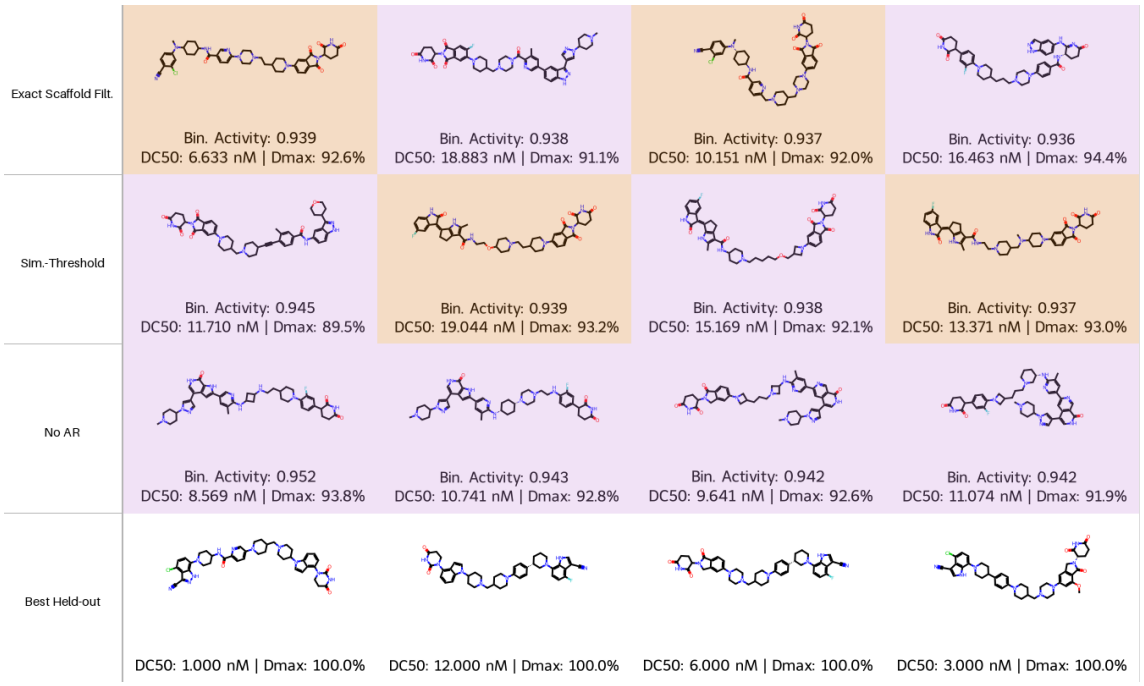


Figure 5.21: Top-4 PROTACs generated by the best REINVENT LSTM models trained on the three different datasets. The best PROTACs are selected based on the predicted binary activity.

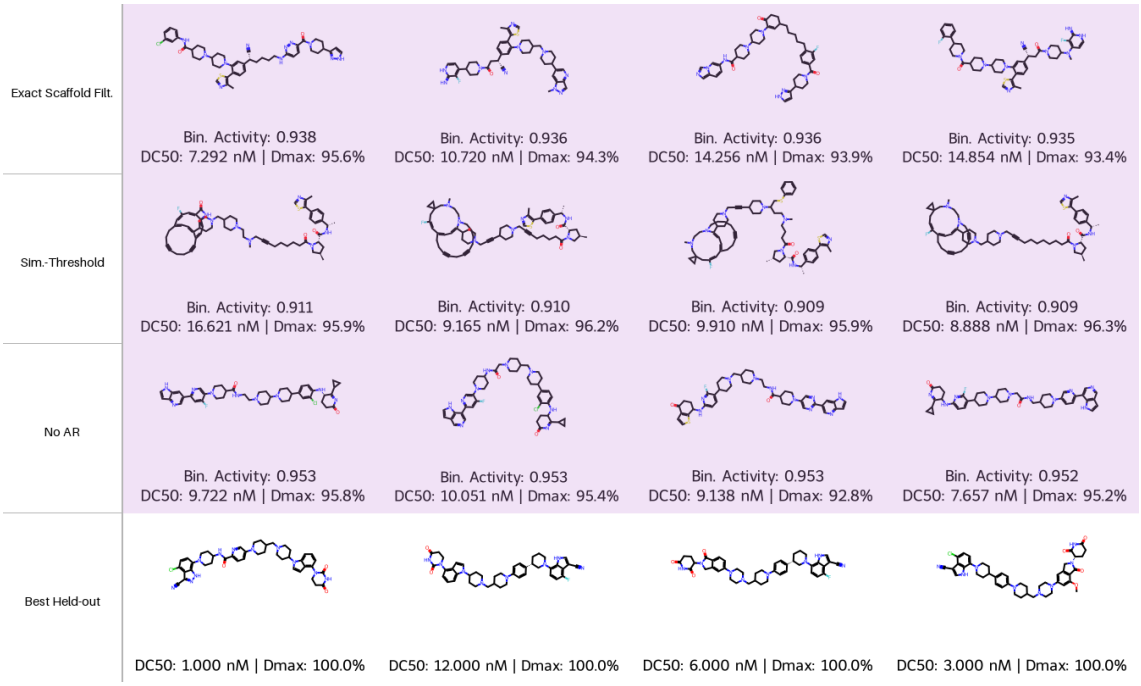


Figure 5.22: Top-4 PROTACs generated by the best Transformer models trained on the three different datasets. The best PROTACs are selected based on the predicted binary activity.

6

Discussion

6.1 Reliability of the Results

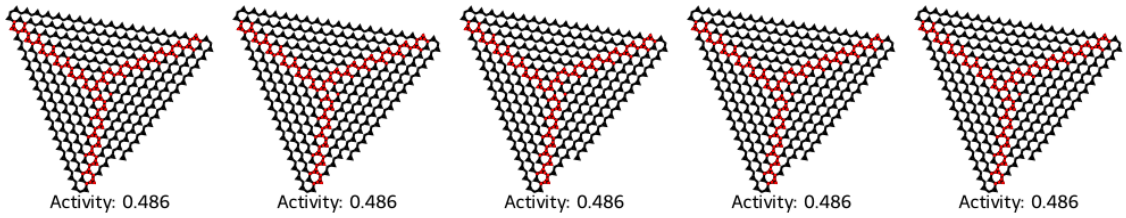
The reliability of the results depends on many factors, and measures have been taken to ensure as reliable results as possible. Firstly, the data processing was developed to achieve as high-quality data as possible for training the models, while also minimizing data leakage. Secondly, the rigorous evaluation pipeline is in place to evaluate both the models and their generated PROTACs against various metrics. However, the TACK surrogate degradation predictor models ultimately determine the quality of the generated PROTACs. This is because the TACK surrogate models are the ones to score the PROTACs according to their degradation capabilities for the target E3, cell line, and POI combination. Therefore, the degradation capability of the generated PROTACs depends on the reliability of the surrogate predictor models. The TACK degradation predictors ensure more reliable predictions by being ensemble models, where for each metric, the average of 25 model predictions is returned as the prediction. This ensures more reliability because the 25 slightly different models affect the prediction of one generated PROTAC. Also, the disagreement of the models quantifies the uncertainty of the prediction. This ensures more reliability, as incorporating the uncertainty in the scoring of PROTACs penalizes uncertain predictions.

As shown in the TACK paper [9], the data used for training the surrogate degradation predictor models consisted mainly of PROTACs with high D_{max} and high pDC_{50} (low DC_{50}) scores. This may make the surrogate models biased towards predicting high D_{max} and pDC_{50}, making them score bad PROTACs too highly. Additionally, this may also result in the model giving high scores to generated PROTACs that are similar to PROTACs known to be active for other E3, cell line, and POI combinations other than the target combination. Specifically, if a generated PROTAC is similar to high-scoring PROTACs for *e.g.,* the IRAK4 POI, the surrogate model might score this generated PROTAC too highly, even though it might not be suitable for the AR POI, which is the target POI. This might be an explanation for why the models rediscover PROTACs for other E3, cell line, and POI combinations than the target combination, as seen in Table 5.3. On the other hand, this may indicate that the rediscovered PROTACs are suitable for the target E3, cell line, and POI combination as well. The only way to confirm this is to synthesize and

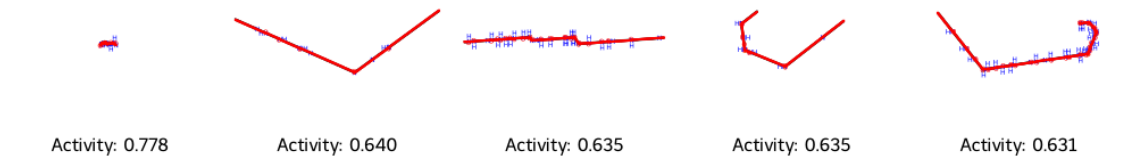
test the PROTACs in a wet lab. Ultimately, the surrogate predictor models are treated as black-box oracles, meaning they are separate from the pipeline and can be substituted. The results show that the pipeline successfully guides the PROTAC generation using RL with surrogate degradation predictor models, showing that the pipeline itself is robust and works as expected.

6.1.1 Reward Hacking

Reward hacking is when, during RL, a model learns to cheat the reward system by outputting things that are nonsense but still yield high rewards. Figure 6.1 presents two different occasions where the Transformer-based model learnt to hack the rewards by generating compounds that are chemically valid and achieve high binary activity scores, but look nothing like PROTACs. In these scenarios, the Transformer-based model has learnt to cheat the binary degradation predictor surrogate model. This might have been made possible by, as previously mentioned, the fact that the TACK degradation predictor models were trained mainly on PROTACs with high D_{max} and pDC_{50} scores, making it give high scores to compounds it has no experience with. Furthermore, introducing a -1 reward for invalid and duplicate SMILES could facilitate reward hacking. This is because the rewards are normalized to be between [0,1], hence a reward signal of -1 is very large and might cause the Transformer to either overly optimize validity and uniqueness or cheat them.



(a)



(b)

Figure 6.1: The figures show two separate occasions on which the Transformer-based model has generated nonsense structures that still got a high activity score by the binary degradation predictor surrogate model.

6.2 Model Generalization Capabilities

Both the REINVENT LSTM and the Transformer model were trained on three different datasets. (1) "Exact Scaffold Filtered" where SMILES with an exact scaf-

fold match to the held-out dataset were removed, (2) "Similarity Threshold" where SMILES with a Tanimoto similarity (Morgan fingerprint radius 8 bit-size 1024) above 0.4 were removed, and (3) "No AR" which is the similarity threshold dataset, but with all SMILES connected to the AR POI removed. This was a generalization test evaluating how well the models learned to generate optimized PROTACs for the target POI, E3, and cell line under three increasingly difficult conditions. Table B.1 and Figures 5.1 and 5.2 display the mean ± 1 standard deviation of the means for 5 runs evaluated against the V.U.N., diversity, and average SA score metrics. These runs utilized identical hyperparameters, which were found to be best-performing for the REINVENT LSTM and the Transformer across the three different datasets. The values were calculated from 10,000 sampled SMILES after completion of each training stage. Figure 5.1 shows that the REINVENT LSTM was mostly stable across almost all metrics with similar means and small standard deviations across the datasets. The uniqueness, diversity, and SA score metrics were not as stable during RL for all three datasets, with the similarity-threshold and the NO AR datasets leading to the most instability. In addition to this, Figure 5.3 plots the mean ± 1 standard deviation of the means for the 5 identical runs over oracle calls during RL training. From these figures, it can be seen that the REINVENT LSTM is the most unstable for the similarity-threshold dataset, since it has a significantly larger standard deviation for that dataset. This is somewhat unexpected since the "No AR" dataset is supposed to be more difficult. The REINVENT LSTM achieves slightly higher DC_{50}, D_{max}, and binary activity for the similarity-threshold dataset but is more unstable. A possible reason for this could be that the similarity-threshold dataset keeps the data that is connected to the AR POI that is most dissimilar to the held-out. Since the held-out dataset is constructed from data used to train the surrogate models, and the held-out dataset consists of high-scoring PROTACs for the target POI, E3, and cell line, the knowledge the model has about the target POI may be hindering more than it is helping. On the other hand, this could also be the result of suboptimal hyperparameters.

Similarly, Figures 5.2 and 5.4 show the evaluation metrics and the sample efficiency for the Transformer model. The Transformer behaves similarly to the REINVENT LSTM but is more unstable. The Transformer is especially unstable during RL, indicated by the fact that it has larger variance during RL for most metrics in 5.2. This means that GRPO introduces this variance, and it could be the result of the additional scoring functions or other configuration choices. Furthermore, as this is the variance between the mean of 5 identical runs, it means that for the same random seed and hyperparameters, the runs still vary significantly. This may be since even though they use the same random seed, GRPO by default sets its parameter **full_determinism** = False, indicating that there is enforced diversity and randomness. Additionally, the Transformer achieves similar scores to the REINVENT LSTM but has slightly worse validity. Since the Transformer is explicitly optimized for validity during RL, either the scoring is suboptimal, or validity is harder to learn for GRPO than for the REINVENT RL strategy.

As shown in 5.4, the Transformer converges at slightly higher degradation rewards

than the REINVENT LSTM, but is instead the most unstable for the exact scaffold filtered dataset, which is expected to be the most stable. This could be the result of suboptimal hyperparameters or perhaps that the model is too general, such that the search space is too large. This, in combination with the fact that some generated PROTACs not targeting the target POI get high scores, could result in unstable learning. Thus, making the model unable to find better optima.

Figures 5.17 and 5.18 show the batch average Tanimoto similarity to the held-out dataset over RL training. The REINVENT LSTM gets slightly similar to the held-out set over RL. This indicates that the REINVENT LSTM is guided slightly closer to known high-scoring PROTACs. On the other hand, the Transformer does not generate PROTACs more similar to the held-out dataset over RL training. This means that it either found a new and different area of the PROTAC space that is promising or that it is reward hacking. Only testing in a wet lab can decide this.

6.2.1 Best Models

Figures 5.9, 5.10, 5.15, and 5.16 show the evaluation metrics and the sample efficiency for the best of the 5 identical runs for the LSTM and the Transformer, respectively. The plots show the batch-average reward ± 1 standard deviation, being the standard deviation within each batch, which shows the spread within each batch of generated PROTACs. The plots show that the variance is relatively high, which is to be expected since the models both optimize for diversity, which forces them to explore. This could lead to a larger variance in the quality of the generated PROTACs.

Figures 5.11 and 5.12 show the distribution of DC_{50} and D_{max} scores of the 10,000 sampled (generated) smiles after each training stage. For both models, the distribution clearly shifts between the stages, becoming increasingly denser. RL results in the biggest shift as expected since that stage optimizes for DC_{50} and D_{max}. This indicates that RL is successful in optimizing both models for DC_{50} and D_{max}, as it shifts both distributions into a denser, higher-scoring area. Figure 5.12 shows that the Transformer got a more aggressive shift during RL than the LSTM, making it have a much denser distribution concentrated around higher DC_{50} and D_{max} scores. This could indicate that the Transformer has a more aggressive RL strategy and learns more efficiently. It could also be the result of reward hacking, as the plots show only the distribution of DC_{50} and D_{max}. Nevertheless, the Transformer finds a highly dense, high-scoring area of the PROTAC space after RL.

Figures 5.13 and 5.14 show the Pareto front for the best run of both models, respectively, across the datasets. The Pareto fronts show the tradeoff between pDC_{50} and D_{max} as it plots the points not dominated by any other point in both pDC_{50} and D_{max}. The Pareto fronts show that for both models and all datasets, not a single generated PROTAC is best at both pDC_{50} and D_{max}. It shows that in order to optimize further for pDC_{50}, D_{max} needs to be sacrificed slightly and vice versa. For the REINVENT LSTM, the run trained on the exact scaffold filtered dataset

generates a PROTAC with significantly higher pDC_{50} than for the other datasets, while the D_{max} is similar for all three. The PROTAC with high pDC_{50} could be a fluke outlier, but could also indicate that the REIVENT LSTM trained on the exact scaffold filtered dataset is able to generate PROTACs with better DC_{50}. For the Transformer, the Pareto fronts for all three datasets are very similar, all achieving similar scores and tradeoffs. For both models, the tradeoff between the datasets is very similar. This could indicate that this is the best possible tradeoff achievable for either the surrogate model, the optimization algorithm, or the scoring function.

6.2.2 Physicochemical Properties Generated PROTACs

During RL, the models were not explicitly trained to optimize for certain physicochemical properties. However, the physicochemical properties of the generated PROTACs are measured as an extra level of evaluation. Figures 5.19 and 5.20 show the distribution of the physicochemical properties of the 10,000 generated PROTACs after RL. They show these distributions for the single best run for both models across all datasets. For most of the metrics, the mean of the generated PROTACs for both models across all datasets lies within the rule of oral PROTACs. This indicates that the fully trained models generate compounds with PROTAC-like properties. Specifically, this indicates that the RL does not make the models divert too much from the curated PROTAC space. However, for both models, LogP and number of rotatable bonds are the physicochemical properties that stray the most out of the RoP boundary. This may be due to either RL or to the compounds in the training data potentially having higher LogP on average.

6.2.3 Best Generated PROTACs

Section 5.2.5 presents the top-4 PROTACs with the highest binary activity score generated by both models after RL for all datasets. Additionally, the 4 PROTACs with the highest D_{max} from the held-out dataset are presented as a comparison. Figure 5.21 presents the top-4 generated PROTACs for the REINVENT LSTM trained on the three datasets. The image shows that all these generated PROTACs are novel, but that the models trained on the exact scaffold filtered and the similarity-threshold datasets have two PROTACs that are similar to the training data. This further indicates that the fully trained models do not divert too much during RL. Since the model trained on the No AR dataset has not seen any AR data during TL, it is expected that all of its top-4 generated PROTACs are novel with low similarity to the training data. All top-4 generated PROTACs for the Transformer trained on all three datasets are novel. As seen in Figure 5.12, the RL for the Transformer shifts its distribution drastically where it finds a dense high-scoring area. This, in combination with the fact that the Transformer does not get more similar to the held-out dataset during RL (Figure 5.18), indicates that it finds a novel space as it generates PROTACs that are not similar to the held-out ones. This novel space yields high predicted degradation scores, meaning either that this is an adequate space for the target POI or that it is reward hacking. Only testing in a wet lab can decide this. However, the generated PROTACs for the Transformer trained on

the similarity-threshold dataset are most likely the result of reward hacking. This is because the large ring structures appearing in the generated compounds are unusual for PROTACs. This can also be seen in Figure 5.10, the synthesizability score (SA Score) for the Transformer model trained on the similarity-threshold dataset increased after RL, whereas it decreased for almost all other RL stages. This indicates that after RL, the generated PROTACs are predicted to be harder to synthesize, emphasizing that the ring structures are undesirable. Nevertheless, both the REINVENT LSTM and the Transformer are capable of generating novel PROTACs that are predicted to be active, having high D_{max} and low DC_{50} scores.

6.3 Data

Machine learning relies heavily on large quantities of high-quality data to produce quality outputs. With more data, the model observes more examples drawn from the underlying target distribution, increasing informed decisions. Data scarcity can therefore be a difficult and recurring problem within machine learning. For the thesis, the TACK dataset offers high-quality PROTACs but is limited in its numbers. Conversely, the synthetic dataset significantly increases the number of training samples but at the cost of a distribution shift compared to the true target distribution. Ultimately, this raises the question of how much data is necessary for the fulfillment of predefined criteria and if the introduction of synthetic data improves the model's ability to efficiently learn the PROTAC space.

Data scarcity poses a challenge both when it comes to total data count and the in-data distribution of DC_{50} and D_{max}. As previously mentioned, only 1054 PROTACs are active, and 50% of those are put aside for the held-out sets. As the goal of the thesis is to generate PROTACs with as low DC_{50} and as high D_{max} as possible, this increases the difficulty of the RL. Training on a larger set of active PROTACs would encourage the model to understand the structure of potent PROTACs during TL. Instead, when the number of active PROTACs is very limited, the model has to rely increasingly on the RL algorithm to guide generation into high-scoring regions, as most training examples are less potent. This has two potential drawbacks. First, the sample efficiency of the model likely decreases as the starting point for RL optimization is based on a worse model. Learning from active PROTACs can be seen as using a more informed prior to the RL. The model will otherwise have to discover the chemical space of active PROTACs rather than using its previously obtained knowledge. Secondly, this can also result in suboptimal model performance and increase the risk of reward hacking. A well-defined model prior might require less tuning to achieve satisfactory performance. Thus, the risk of reward hacking can be reduced as large deviations from the prior's sample space are not necessary.

6.4 The Effect of Staged Learning

The reasoning behind employing staged learning is to stabilize learning by gradually teaching the models to generate PROTACs with optimal properties. This is

because tasking a prior model trained on SMILES of small molecules to generate PROTACs with multiple optimal properties might be too complex to learn simultaneously. Therefore, staged learning is suitable in order to break down the task into less complex steps by first teaching the model the syntax of PROTAC-like molecules, then the syntax of real, high-scoring PROTACs, and finally guiding it towards PROTACs with optimal DC_{50} and D_{max}. The result from this can be seen in Figure 5.11. By leveraging TL and RL in stages, the distribution of generated PROTACs can be shifted gradually towards high-scoring PROTACs in terms of DC_{50} and D_{max}. Another reason to use staged learning is the data scarcity of curated PROTACs. Since only a small number of curated PROTACs exist, the data is too scarce to efficiently train a large ML model. Therefore, by leveraging a large set of synthetic PROTAC-like data, the model can get familiar with PROTAC syntax. Thus, the model does not need to learn this from the small curated PROTACs dataset. Instead, it can more efficiently learn the rules of curated PROTACs.

A noticeable result that can be seen in Table B.1 is that the validity of every model drops after being trained on the curated PROTAC data. This is most likely a consequence of the small size of the curated PROTAC dataset. Since the space of the curated PROTACs is different from the space of the synthetic PROTACs, the model's distribution is shifted closer to the curated PROTAC space when training. Therefore, the model needs to break some of its previous knowledge, resulting in a drop in validity. However, since the curated PROTAC dataset is so small, the model is unable to recover its validity quickly enough before overfitting. Overfitting makes the model remember its training data too strongly rather than generalizing from the patterns in the training data. Consequently, it is favorable to choose models with lower validity and no overfitting. Hence, the best models after fine-tuning on the curated PROTAC data have lower validity but better generalization capabilities. As can also be seen in Table B.1, the RL also optimizes for validity, making the models recover their validity without overfitting on the curated PROTAC data.

6.5 Fingerprint Construction

The fingerprint radius and bit size can greatly influence the differences in performance between datasets. Increasing the radius of the fingerprint increases the number of neighbors taken into account when assigning bits for each atom. As such, small differences in structures have a larger effect on the fingerprint if the radius is larger. Conversely, reducing the radius reduces the effect of global structures for each bit. An example of this can be seen when comparing figures 4.1 and 6.2. Notably, the molecule with Tanimoto similarity of 0.58 in figure 4.1 is visually similar to the reference. It is also assigned a higher score of 0.78 when using a radius of 3 as opposed to radius 8, as seen in figure 6.2. Likewise, the molecule assigned a Tanimoto similarity of 0.2 in figure 6.2 is more distinct than the reference when compared to the molecule assigned 0.2 in figure 4.1. Ultimately, this means that the radius influences the similarity assigned to each model. Both the similarity-threshold dataset and the No AR dataset used a Tanimoto similarity filter of 0.4 to. As such, using a larger radius preserves similar molecules to a higher degree than a

smaller radius. This could be an explanation for why there isn't a large difference in performance between the exact scaffold filtered dataset and the similarity-threshold dataset. A radius of 8 might preserve enough information in the similarity-threshold for the difficulty to be consistent.

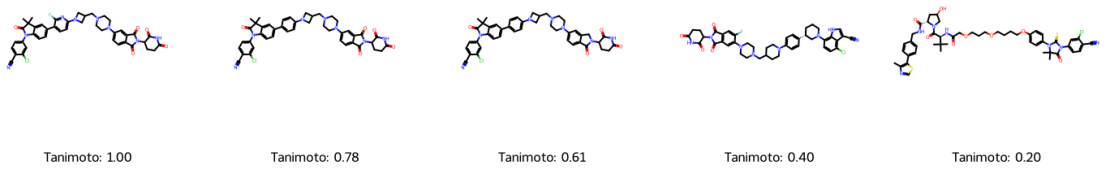


Figure 6.2: The figure shows Tanimoto similarity between molecules using a radius of 3 and bit-size of 1024

6.6 Comparing the REINVENT LSTM and the Transformer

Both the LSTM introduced in REINVENT4 [6], and the Transformer-based NovoMolGen [10] are considered state-of-the-art models for molecular design. Chitsaz *et al.* compared evaluation metrics directly to the REINVENT counterpart; however, for this thesis, a direct comparison between the two is possibly unfair. On their own, the LSTM model and the Transformer model can indicate the capability of generating valid, high-scoring PROTACs. However, fundamental differences in the pre-training, parameter count, and training algorithms hinder a fair comparison of the PROTACs generated by the LSTM and the Transformer model. The aforementioned LSTM was pre-trained on a substantially smaller set of molecules (119M) compared to the 1.5B molecules used to train the Transformer model. Additionally, the smaller dataset did not constitute a direct subset of the Transformer's much larger dataset. Therefore, comparing the LSTM's ability to shift generation to the PROTAC space through synthetic TL cannot be compared directly to the Transformer's. This is due to the fact that the pre-trained LSTM version's dataset might be either more or less similar compared to the Transformer's. Thus, converging differently.

There is a substantial difference in the number of trainable parameters for the models. As aforementioned, the LSTM has 22M tunable parameters, whereas the Transformer has 32M. This means that the Transformer model has roughly 45.4% more parameters than the LSTM. A difference in the number of parameters ultimately influences a model's ability to capture and learn complex patterns along with fine-grained details. Kaplan *et al.* [72] showed that model performance scales sub-linearly for Transformer-based language models if the number of parameters is increased for a constant dataset. Although Kaplan *et al.* developed the scaling laws for language models, their insights can still be partly transferred to molecular generation as both the LSTM and the Transformer generate molecules using text outputs. According to

the scaling laws, diminishing returns in terms of performance can be seen as the number of parameters that are scaled for a fixed dataset size. However, the performance should be no worse than a model with fewer parameters. This is due to the fact that a network with more parameters could ultimately inactivate parameters to collapse to the lesser variant. On the contrary, more parameters do, however, increase the difficulty in preventing training data memorization. Therefore, it is difficult to compare the architectural aspects of generated PROTACs fairly without fixing the same number of tunable parameters for both the LSTM and the Transformer.

Finally, REINVENT uses an alternative optimization algorithm and targets optimization criteria differently from the explicit criteria defined for the Transformer. Molecules that are invalid are assigned a score of 0; thus, REINVENT implicitly optimizes for validity during RL even if it is not explicitly defined. Furthermore, the Transformer model uses GRPO and implements the diversity penalty differently than REINVENT's bucket method. All of the previous differences indicate that conclusions about the final performance cannot be drawn based on the network architecture used. Instead, notable differences in performance could be caused by the differences in pre-training, parameter count, and the goal-directed optimization through RL. On their own, the models can instead indicate their ability to direct generation into high-scoring PROTAC areas of the chemical space.

6.7 Sample Efficiency

As previously mentioned, academia rarely presents sample efficiency for more than 10,000 oracle calls. Both the LSTM and the Transformer can be seen to require around 30,000 oracle calls before converging. This is significantly more than the standard upper boundary; however, oracle calls to the surrogate model are cheap compared to other oracle calls. Therefore, the difference between 10,000 and 30,000 oracle calls is not as significant as for a more expensive oracle. There are several aspects that can influence the number of oracle calls required for convergence, including sample batch size for each RL step, the number of optimization criteria used, and the size of the molecule to be optimized.

Large molecules, such as PROTACs, increase the difficulty of optimization due to the number of available modification sites. As aforementioned, the linker region of the PROTAC is especially flexible, which can result in the optimization of PROTACs requiring additional oracle calls compared to optimization for small inhibitor drug design. The sample efficiency could likely be reduced by decreasing the number of PROTACs sampled for each RL step. However, doing so increases the noise and randomness of the system. A larger batch size stabilizes training by improving the accuracy of the gradient used to update the model since a larger sample size better represents the ideal direction for optimization. Finally, the number of optimization criteria and the stability of the oracle used can influence the sample efficiency immensely. With more optimization criteria, the complexity of the optimization task increases. Noisy predictions from surrogate models can also hinder fast convergence due to conflicting directions of optimization for similar molecules.

6.8 Propagation of Error Through the Pipeline

The pipeline was created to gradually teach the model to generate potent PROTACs. However, the multi-stage design can also be a cause of suboptimal performance after RL. Errors such as a poorly trained model during TL will propagate throughout the system, affecting the subsequent training steps before the final impact of it is known. The prior for the RL is based on the best model from TL, thus the starting point of RL in the chemical space is determined by the previous model. Said starting point will greatly influence the final outcome, limiting exploration for PROTAC generation or guiding generation towards suboptimal areas of the chemical space. As such, there is an emphasis on creating good models at each stage of the training as those are used for the subsequent stages.

6.9 Comparison to Related Works

As mentioned previously, Nori *et al.* [8] optimized GraphINVENT, a graph-based PROTAC generation model, for DC_{50} using a boosted-tree-based binary activity predictor surrogate model. Specifically, their surrogate model predicts a PROTAC to be active if its $DC_{50} < 100$ nM. Nori *et al.* applied policy gradient RL and used the surrogate predictor in their scoring. After RL training, they sampled 10,000 PROTACs of which 82% were predicted active ($DC_{50} < 100$ nM), and close to 100% were chemically valid and novel. Furthermore, in their paper, Nori *et al.* targeted the IRAK3 POI. These results are not directly comparable to those of this thesis due to differences in target POI, training data, optimization targets, and the definition of activity. However, as seen in Table 5.2, out of 10,000 sampled PROTACs, the REINVENT LSTM trained on the three different datasets has 9217 - 9255 valid, unique, and canonicalized SMILES. For those the model achieves 96 - 98% with $DC_{50} < 100$ nM, 86 - 889% with $D_{max} > 80\%$, and 90 - 92% predicted active by the binary predictor. Furthermore, out of 10,000 sampled PROTACs, the Transformer model trained on the three different datasets has 7755 - 9112 valid, unique, and canonicalized SMILES. Out of those, 99 - 100% have $DC_{50} < 100$ nM, 99 - 100% have $D_{max} > 80\%$, and 95 - 99% are predicted active by the binary predictor. Although no conclusion can be drawn regarding whether the results of this thesis outperform Nori *et al.* or not, they nevertheless fall within a similar range to what is reported in modern research.

7

Future Work

The thesis has focused on generating PROTACs for the AR POI along with the CRBN ligase. With promising results, further research could target new areas of PROTAC generations, including other POIs such as BTK, EGFR, STAT3, and extend the E3 ligase targets to include the widely used VHL. This would strengthen the work further by providing additional insights into the robustness and the versatility of the generation pipeline. Currently, the pipeline only supports a single pair of POI and E3 as a target. Adjusting to new targets is not difficult but time-consuming, as it would require retraining of all steps in the pipeline, including the TL. The aforementioned TL is used to shift generation into the PROTAC space. The training procedure of TL does not rely on a specific POI and E3 pair, but the evaluation does. Evaluation of the best model at each step incorporates properties such as DC_{50}, D_{max}, and binary activity predicted by the surrogate model. As such, model selection has a preference bias towards the targeted AR and CRBN. Therefore, changing the targets would require complete retraining and evaluation. An interesting research direction for multi-target PROTAC design would be to explore and create a foundation model capable of generating a diverse set of PROTACs with different target POIs and E3s. This could be done in one of two ways: (1) use an encoder-decoder Transformer to encode information about the targets, or (2) use a mixture of experts model.

The current implementation of RL requires more oracle calls to achieve satisfactory results than the widely adopted 10,000 oracle calls presented in academia for molecular generation. Currently, around 30,000 oracle calls are required for convergence, both for the LSTM and for the Transformer model. Improving the sample efficiency would likely be a prioritized area of future work in order to adhere to the academic norm. Doing so would also improve the usability of the pipeline, as less computation would be required for potent PROTAC generation. As previously mentioned, there are several factors influencing the sample efficiency. Thus, research would likely focus on further refinement of hyperparameter selection, along with a study focusing on the effect of different optimization criteria for the sample efficiency.

Analyzing different optimization criteria could not only be beneficial in increasing the sample efficiency but also the performance of the entire system. The current reward signal used during RL primarily uses DC_{50}, D_{max}, and binary activity, along

with indirect or direct optimization of validity and diversity. Scores such as the synthesizability could be incorporated into the optimization to guide generation towards regions with PROTACs that are feasible to create. The current implementation of RL implicitly optimizes for PROTAC properties such as molecular weight, due to its training data consisting of curated PROTACs [9]. However, this could be explicitly added as an optimization criterion to provide more rigorous guidance for the RL algorithm.

Finally, the unconditional nature of the generation pipeline could be leveraged to create novel datasets of synthetic but realistic PROTACs. By creating such a dataset, the gap between real PROTACs and synthetic ones could be reduced to increase performance further. In order to release such a dataset, additional work would be required to evaluate the outcome of the models, filter generated PROTACs that are not consistent with PROTAC properties, and ensure diverse generation for multiple POIs and E3s. Releasing a large synthetic dataset of realistic PROTAC could enable pretraining of foundation models for PROTAC generation, ultimately advancing the field of PROTAC research further.

8

Conclusion

The thesis investigated the effect of reinforcement learning for guiding the generation of PROTACs. More specifically, generation was guided, and degradation was predicted for the AR protein using the LNCaP cell line and the CRBN E3 ligase. A three-step training procedure comprising synthetic TL, curated TL, and RL was employed to gradually improve generation capability and simultaneously address the data scarcity of curated PROTACs. Two different model architectures were used, an LSTM building on the REINVENT4 [6] ecosystem and a flash-attention Transformer from NovoMolGen [10]. Finally, a total of three different datasets were created to assess how well reinforcement learning is able to guide generation toward the AR target for different starting points.

The LSTM model achieved similar performance across all datasets, indicating the effectiveness of reinforcement learning. The LSTM managed to increase the Tanimoto similarity to the held-out set through RL, however, not by a substantial amount. Instead, the LSTM could be seen to shift the sample space for PROTACs to preferred regions in terms of potency and degradation ability. A new warhead was discovered during reinforcement learning with D_{max} at 93.8%, DC_{50} at 8.57nM, and a predicted binary activity of 95.2%. Moreover, for the V.U.N metrics, the LSTM reached 97.9%, 98.9%, 99.8% respectively on the largest dataset. Structurally, the generated PROTACs shared similarities with known and curated PROTACs without direct optimization targeting physicochemical properties.

Reinforcement learning was further used to guide molecular generation for the Transformer, targeting regions of the chemical space with highly efficient PROTACs capable of degrading the AR protein. For the Transformer, generated PROTACs reached a maximum degradation efficiency of 97.5% and pDC_{50} of 7.99. Additionally, the Transformer achieved 95.6% validity, 99.1% uniqueness, and 100% novelty on the largest dataset. Generated PROTACs could, however, be seen to differ structurally compared to the held-out set. Indicating that the Transformer successfully found a new region of PROTACs worth exploring or that the model was able to reward hack.

In conclusion, results gathered for both models indicated that the reinforcement learning using the surrogate model rewards is effective in guiding the generation

towards the AR target. Furthermore, novel potential PROTAC candidates were discovered. However, wet-lab testing is required to verify the true degradation capability of the PROTAC candidates.

Bibliography

[1] K. Li and C. M. Crews, "Protacs: Past, present and future," *Chemical Society Reviews*, vol. 51, no. 12, pp. 5214–5236, 2022. DOI: https://doi.org/10.1039/d2cs00193d.

[2] Z. Hu and C. M. Crews, "Recent Developments in PROTAC-Mediated Protein Degradation: From Bench to Clinic," *ChemBioChem*, vol. 23, no. 2, e202100270, 2022. DOI: https://doi.org/10.1002/cbic.202100270. eprint: https://chemistry-europe.onlinelibrary.wiley.com/doi/pdf/10.1002/cbic.202100270. [Online]. Available: https://chemistry-europe.onlinelibrary.wiley.com/doi/abs/10.1002/cbic.202100270.

[3] G. M. Burslem and C. M. Crews, "Proteolysis-targeting chimeras as therapeutics and tools for biological discovery," *Cell*, vol. 181, no. 1, pp. 102–110, 2019. DOI: 10.1016/j.cell.2019.11.031. [Online]. Available: https://doi.org/10.1016/j.cell.2019.11.031.

[4] M. Toure and C. M. Crews, "Small-molecule protacs: New approaches to protein degradation," *Angewandte Chemie International Edition*, vol. 55, no. 6, pp. 1966–1973, 2016. DOI: 10.1002/anie.201507978.

[5] I. Churcher, "Protac-induced protein degradation in drug discovery: Breaking the rules or just making new ones?" *Journal of Medicinal Chemistry*, vol. 61, pp. 444–452, 2 Jan. 2018, ISSN: 15204804. DOI: 10.1021/acs.jmedchem.7b01272.

[6] H. H. Loeffler et al., "Reinvent 4: Modern aidriven generative molecule design," *Journal of Cheminformatics*, vol. 16, no. 1, 2024, ISSN: 1758-2946. DOI: 10.1186/s13321-024-00812-5. [Online]. Available: https://doi.org/10.1186/s13321-024-00812-5.

[7] S. Zheng et al., "Accelerated rational protac design via deep learning and molecular simulations," *Nature Machine Intelligence*, vol. 4, pp. 739–748, 9 2022, ISSN: 2522-5839. DOI: 10.1038/s42256-022-00527-y. [Online]. Available: https://doi.org/10.1038/s42256-022-00527-y.

[8] D. Nori, C. W. Coley, and R. Mercado, *De novo protac design using graph-based deep generative models*, 2022. arXiv: 2211.02660 [q-bio.QM]. [Online]. Available: https://arxiv.org/abs/2211.02660.

[9] S. Ribes, N. Dunlop, and R. Mercado, *Tack: A statistical evaluation of degradation activity on a novel targeting chimeras knowledge dataset*, 2026. arXiv: 2605.19579 [q-bio.QM]. [Online]. Available: https://arxiv.org/abs/2605.19579.

[10] K. Chitsaz, R. Balaji, Q. Fournier, N. P. Bhatt, and S. Chandar, "Novomolgen: Rethinking molecular language model pretraining," in *ICML 2025 Generative AI and Biology (GenBio) Workshop*, 2025. [Online]. Available: https://openreview.net/forum?id=5OPItDY07C.

[11] V. Fialková et al., "Libinvent: Reaction-based generative scaffold decoration for in silico library design," *Journal of Chemical Information and Modeling*, vol. 62, no. 9, pp. 2046–2063, 2022, ISSN: 1549-9596. DOI: 10.1021/acs.jcim.1c00469. [Online]. Available: https://doi.org/10.1021/acs.jcim.1c00469.

[12] J. Guo et al., "Link-invent: Generative linker design with reinforcement learning," *Digital Discovery*, vol. 2, pp. 392–408, 2 2023. DOI: 10.1039/D2DD00115B. [Online]. Available: http://dx.doi.org/10.1039/D2DD00115B.

[13] J. He et al., "Transformer-based molecular optimization beyond matched molecular pairs," *Journal of Cheminformatics*, vol. 14, no. 1, p. 18, 2022, ISSN: 1758-2946. DOI: 10.1186/s13321-022-00599-3. [Online]. Available: https://doi.org/10.1186/s13321-022-00599-3.

[14] J. He et al., "Molecular optimization by capturing chemists intuition using deep neural networks," *Journal of Cheminformatics*, vol. 13, no. 1, p. 26, Mar. 20, 2021, ISSN: 1758-2946. DOI: 10.1186/s13321-021-00497-0. [Online]. Available: https://doi.org/10.1186/s13321-021-00497-0.

[15] M. Olivecrona, T. Blaschke, O. Engkvist, and H. Chen, "Molecular de-novo design through deep reinforcement learning," *Journal of Cheminformatics*, vol. 9, no. 1, p. 48, 2017, ISSN: 1758-2946. DOI: 10.1186/s13321-017-0235-x. [Online]. Available: https://doi.org/10.1186/s13321-017-0235-x.

[16] T. Blaschke et al., "Reinvent 2.0: An ai tool for de novo drug design," *Journal of Chemical Information and Modeling*, vol. 60, no. 12, pp. 5918–5922, 2020, ISSN: 1549-9596. DOI: 10.1021/acs.jcim.0c00915. [Online]. Available: https://doi.org/10.1021/acs.jcim.0c00915.

[17] P. Formont, M. Darrin, I. B. Ayed, and P. Piantanida, "Molrgen: A training and evaluation setting for de novo molecular generation with reasonning models," 2026. arXiv: 2603.18256 [cs.LG]. [Online]. Available: https://arxiv.org/abs/2603.18256.

[18] Z. Shao et al., "Deepseekmath: Pushing the limits of mathematical reasoning in open language models," 2024. arXiv: 2402.03300 [cs.CL]. [Online]. Available: https://arxiv.org/abs/2402.03300.

[19] S. Ribes et al., "Protac-splitter: A machine learning framework for automated identification of protac substructures," *ChemRxiv*, vol. 2025, no. 0708, 2025. DOI: 10.26434/chemrxiv-2025-bn1nv. eprint: https://chemrxiv.org/doi/pdf/10.26434/chemrxiv-2025-bn1nv. [Online]. Available: https://chemrxiv.org/doi/abs/10.26434/chemrxiv-2025-bn1nv.

[20] S. Ribes, E. Nittinger, C. Tyrchan, and R. Mercado, "Modeling protac degradation activity with machine learning," *Artificial Intelligence in the Life Sciences*, vol. 6, pp. 100–104, 2024, ISSN: 2667-3185. DOI: https://doi.org/10.1016/j.ailsci.2024.100104. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2667318524000114.

[21] B. Li, T. Ran, and H. Chen, "3d based generative protac linker design with reinforcement learning," *Briefings in Bioinformatics*, vol. 24, no. 5, bbad323, Sep. 2023, ISSN: 1477-4054. DOI: 10.1093/bib/bbad323. eprint: https://academic.oup.com/bib/article-pdf/24/5/bbad323/51711826/bbad323.pdf. [Online]. Available: https://doi.org/10.1093/bib/bbad323.

[22] F. Imrie, A. R. Bradley, M. van der Schaar, and C. M. Deane, "Deep generative models for 3d linker design," *Journal of Chemical Information and Modeling*, vol. 60, no. 4, pp. 1983–1995, Apr. 27, 2020, ISSN: 1549-9596. DOI: 10.1021/acs.jcim.9b01120. [Online]. Available: https://doi.org/10.1021/acs.jcim.9b01120.

[23] Q. Hu et al., "Deepdegradome: A structure-aware deep learning framework for protac and ligand generation against protein targets," *Proceedings of the National Academy of Sciences*, vol. 123, no. 11, e2518248123, 2026. DOI: 10.1073/pnas.2518248123. eprint: https://www.pnas.org/doi/pdf/10.1073/pnas.2518248123. [Online]. Available: https://www.pnas.org/doi/abs/10.1073/pnas.2518248123.

[24] F. Li et al., "Deepprotacs is a deep learning-based targeted degradation predictor for protacs," *Nature Communications*, vol. 13, no. 1, p. 7133, 2022, ISSN: 2041-1723. DOI: 10.1038/s41467-022-34807-3. [Online]. Available: https://doi.org/10.1038/s41467-022-34807-3.

[25] M. Toure and C. M. Crews, "Small-Molecule PROTACS: New Approaches to Protein Degradation," *Angewandte Chemie International Edition*, vol. 55, no. 6, pp. 1966–1973, 2016. DOI: https://doi.org/10.1002/anie.201507978. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/anie.201507978. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/anie.201507978.

[26] J. Ge et al., "Protac-db 3.0: An updated database of protacs with extended pharmacokinetic parameters," *Nucleic Acids Research*, vol. 53, no. D1, pp. D1510–D1515, Sep. 2024, ISSN: 1362-4962. DOI: 10.1093/nar/gkae768. eprint: https://academic.oup.com/nar/article-pdf/53/D1/D1510/59000756/gkae768.pdf. [Online]. Available: https://doi.org/10.1093/nar/gkae768.

[27] K. Tanaka, "The proteasome: Overview of structure and functions," *Proc Jpn Acad Ser B Phys Biol Sci*, vol. 85, no. 1, pp. 12–36, 2009.

[28] Y. Chen, I. Tandon, W. Heelan, Y. Wang, W. Tang, and Q. Hu, "Proteolysis-targeting chimera (PROTAC) delivery system: Advancing protein degraders towards clinical translation," *Chem Soc Rev*, vol. 51, no. 13, pp. 5330–5350, Jul. 2022.

[29] 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, 1988, ISSN: 0095-2338. DOI: 10.1021/ci00057a005. [Online]. Available: https://doi.org/10.1021/ci00057a005.

[30] Daylight Theory: SMILES, www.daylight.com, May 19, 2025. [Online]. Available: https://www.daylight.com/dayhtml/doc/theory/theory.smiles.html.

[31] L. David, A. Thakkar, R. Mercado, and O. Engkvist, "Molecular representations in ai-driven drug discovery: A review and practical guide," *Journal of Cheminformatics*, vol. 12, no. 1, p. 56, Sep. 2020, ISSN: 1758-2946. DOI: 10.1186/s13321-020-00460-5. [Online]. Available: https://doi.org/10.1186/s13321-020-00460-5.

[32] D. Rogers and M. Hahn, "Extended-connectivity fingerprints," *Journal of Chemical Information and Modeling*, vol. 50, no. 5, pp. 742–754, May 2010, ISSN: 1549-9596. DOI: 10.1021/ci100050t. [Online]. Available: https://doi.org/10.1021/ci100050t.

[33] 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, ISSN: 0021-9576. DOI: 10.1021/c160017a018. [Online]. Available: https://doi.org/10.1021/c160017a018.

[34] G. Landrum, *RDKit: Open-source cheminformatics*, 2024. [Online]. Available: https://www.rdkit.org.

[35] T. T. Tanimoto, "IBM Internal Report: An Elementary Mathematical Theory of Classification and Prediction," 1957. [Online]. Available: http://dalkescientific.com/tanimoto.pdf.

[36] G. W. Bemis and M. A. Murcko, "The properties of known drugs. 1. molecular frameworks," *Journal of Medicinal Chemistry*, vol. 39, no. 15, pp. 2887–2893, 1996, ISSN: 0022-2623. DOI: 10.1021/jm9602928. [Online]. Available: https://doi.org/10.1021/jm9602928.

[37] C. A. Lipinski, F. Lombardo, B. W. Dominy, and P. J. Feeney, "Experimental and computational approaches to estimate solubility and permeability in drug discovery and development settings," *Advanced Drug Delivery Reviews*, vol. 23, no. 1, pp. 3–25, 1997, In Vitro Models for Selection of Development Candidates, ISSN: 0169-409X. DOI: https://doi.org/10.1016/S0169-409X(96)00423-1. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0169409X96004231.

[38] J. S. Scott, I. N. Michaelides, and M. Schade, "Property-based optimisation of protacs," *RSC Med. Chem.*, vol. 16, pp. 449–456, 2 2025. DOI: 10.1039/D4MD00769G. [Online]. Available: http://dx.doi.org/10.1039/D4MD00769G.

[39] M. Arduengo, "Developing effective degrader compounds: Why cellular degradation kinetics are key," 2025. DOI: se.promega.com/resources/pubhub/2025/developing-effective-degrader-compounds-why-cellular-degradation-kinetics-are-key/.

[40] R. M. Schmidt, *Recurrent neural networks (rnns): A gentle introduction and overview*, 2019. arXiv: 1912.05911 [cs.LG]. [Online]. Available: https://arxiv.org/abs/1912.05911.

[41] A. Graves, *Generating sequences with recurrent neural networks*, 2014. arXiv: 1308.0850 [cs.NE]. [Online]. Available: https://arxiv.org/abs/1308.0850.

[42] I. D. Mienye, T. G. Swart, and G. Obaido, "Recurrent neural networks: A comprehensive review of architectures, variants, and applications," *Information*,

vol. 15, no. 9, 2024, ISSN: 2078-2489. DOI: 10.3390/info15090517. [Online]. Available: https://www.mdpi.com/2078-2489/15/9/517.

[43] S.-H. Noh, "Analysis of gradient vanishing of rnns and performance comparison," *Information*, vol. 12, no. 11, 2021, ISSN: 2078-2489. DOI: 10.3390/info12110442. [Online]. Available: https://www.mdpi.com/2078-2489/12/11/442.

[44] J. F. Kolen and S. C. Kremer, "Gradient flow in recurrent nets: The difficulty of learning longterm dependencies," in *A Field Guide to Dynamical Recurrent Networks*. 2001, pp. 237–243. DOI: 10.1109/9780470544037.ch14.

[45] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997. DOI: 10.1162/neco.1997.9.8.1735.

[46] F. Gers, J. Schmidhuber, and F. Cummins, "Learning to forget: Continual prediction with lstm," in *1999 Ninth International Conference on Artificial Neural Networks ICANN 99. (Conf. Publ. No. 470)*, vol. 2, 1999, 850–855 vol.2. DOI: 10.1049/cp:19991218.

[47] A. Vaswani et al., "Attention is all you need," in *Advances in Neural Information Processing Systems*, I. Guyon et al., Eds., vol. 30, Curran Associates, Inc., 2017. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

[48] C. Meister, T. Vieira, and R. Cotterell, *If beam search is the answer, what was the question?* 2021. arXiv: 2010.02650 [cs.CL]. [Online]. Available: https://arxiv.org/abs/2010.02650.

[49] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Advances in Neural Information Processing Systems*, Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Weinberger, Eds., vol. 27, Curran Associates, Inc., 2014. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2014/file/5a18e133cbf9f257297f410bb7eca942-Paper.pdf.

[50] Z. Fu, W. Lam, A. M.-C. So, and B. Shi, *A theoretical analysis of the repetition problem in text generation*, 2021. arXiv: 2012.14660 [cs.CL]. [Online]. Available: https://arxiv.org/abs/2012.14660.

[51] A. Holtzman, J. Buys, L. Du, M. Forbes, and Y. Choi, *The curious case of neural text degeneration*, 2020. arXiv: 1904.09751 [cs.CL]. [Online]. Available: https://arxiv.org/abs/1904.09751.

[52] F. Zhuang et al., "A comprehensive survey on transfer learning," *Proceedings of the IEEE*, vol. 109, no. 1, pp. 43–76, 2021. DOI: 10.1109/JPROC.2020.3004555.

[53] Q. Yang and S. J. Pan, " A Survey on Transfer Learning," *IEEE Transactions on Knowledge & Data Engineering*, vol. 22, no. 10, pp. 1345–1359, Oct. 2010, ISSN: 1558-2191. DOI: 10.1109/TKDE.2009.191. [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/TKDE.2009.191.

[54] R. S. Sutton and A. G. Barto, Reinforcement Learning: An Introduction, Second edition. The MIT Press, 2015. https://web.stanford.edu/class/psych209/Readings/SuttonBartoIPRLBook2ndEd.pdf.

[55] J. Skalse, N. Howe, D. Krasheninnikov, and D. Krueger, "Defining and characterizing reward gaming," in *Advances in Neural Information Processing Sys-*

tems, S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, Eds., vol. 35, Curran Associates, Inc., 2022, pp. 9460–9471. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2022/file/3d719fee332caa23d5038b8a90e81796-Paper-Conference.pdf.

[56] J. Clark and D. Amodei, Faulty reward functions in the wild, Openai.com, 2016. https://openai.com/index/faulty-reward-functions/.

[57] A. Plaat, "Deep reinforcement learning, a textbook," *CoRR*, vol. abs/2201.02135, 2022. arXiv: 2201.02135. [Online]. Available: https://arxiv.org/abs/2201.02135.

[58] J. Shlens, "Notes on kullback-leibler divergence and likelihood," 2014. arXiv: 1404.2000 [cs.IT]. [Online]. Available: https://arxiv.org/abs/1404.2000.

[59] M. Nagy, Y. Mansour, and S. Abdelmohsen, "Multi-objective optimization methods as a decision making strategy," *International Journal of Engineering and Technical Research*, vol. 9, pp. 516–522, Mar. 2020. DOI: 10.17577/IJERTV9IS030480. [Online]. Available: https://www.ijert.org/multi-objective-optimization-methods-as-a-decision-making-strategy.

[60] C.-L. Hwang and K. Yoon, *Multiple Attribute Decision Making*. Springer Berlin Heidelberg, 1981. DOI: 10.1007/978-3-642-48318-9. [Online]. Available: https://link.springer.com/book/10.1007/978-3-642-48318-9.

[61] I. Pletnev, A. Erin, A. McNaught, K. Blinov, D. Tchekhovskoi, and S. Heller, "Inchikey collision resistance: An experimental testing," *Journal of Cheminformatics*, vol. 4, Dec. 2012. DOI: 10.1186/1758-2946-4-39. Accessed: Apr. 21, 2026.

[62] T. Wolf et al., *Huggingface's transformers: State-of-the-art natural language processing*, 2020. arXiv: 1910.03771 [cs.CL]. [Online]. Available: https://arxiv.org/abs/1910.03771.

[63] L. von Werra et al., *TRL: Transformers Reinforcement Learning*, version 1.3, Mar. 2020. [Online]. Available: https://github.com/huggingface/trl.

[64] S. Kim et al., "Pubchem 2025 update," *Nucleic Acids Research*, vol. 53, no. D1, pp. D1516–D1525, Jan. 2025, ISSN: 1362-4962. DOI: 10.1093/nar/gkae1059. eprint: https://academic.oup.com/nar/article-pdf/53/D1/D1516/60743708/gkae1059.pdf. [Online]. Available: https://doi.org/10.1093/nar/gkae1059.

[65] A. Grattafiori et al., *The llama 3 herd of models*, 2024. arXiv: 2407.21783 [cs.AI]. [Online]. Available: https://arxiv.org/abs/2407.21783.

[66] T. Dao, D. Y. Fu, S. Ermon, A. Rudra, and C. Ré, *Flashattention: Fast and memory-efficient exact attention with io-awareness*, 2022. arXiv: 2205.14135 [cs.LG]. [Online]. Available: https://arxiv.org/abs/2205.14135.

[67] T. Coste, U. Anwar, R. Kirk, and D. Krueger, *Reward model ensembles help mitigate overoptimization*, 2024. arXiv: 2310.02743 [cs.LG]. [Online]. Available: https://arxiv.org/abs/2310.02743.

[68] S. Betala et al., *Lemat-genbench: A unified evaluation framework for crystal generative models*, 2026. arXiv: 2512.04562 [cs.LG]. [Online]. Available: https://arxiv.org/abs/2512.04562.

[69] Y. Zhang, Z. Zhang, L. Wen, and Y. Liu, "Synthetic accessibility scoring and its application to the high-throughput design of energetic molecules," *J. Mater. Chem. A*, vol. 13, pp. 16 330–16 344, 22 2025. DOI: 10.1039/D5TA01042J. [Online]. Available: http://dx.doi.org/10.1039/D5TA01042J.

[70] X. Han et al., "Discovery of highly potent and efficient protac degraders of androgen receptor (ar) by employing weak binding affinity vhl e3 ligase ligands," *Journal of Medicinal Chemistry*, vol. 62, no. 24, pp. 11 218–11 231, 2019, ISSN: 0022-2623. DOI: 10.1021/acs.jmedchem.9b01393. [Online]. Available: https://doi.org/10.1021/acs.jmedchem.9b01393.

[71] K.-H. Chang, "Chapter 5 - multiobjective optimization and advanced topics," in *Design Theory and Methods Using CAD/CAE*, Boston: Academic Press, 2015, pp. 325–406, ISBN: 978-0-12-398512-5. DOI: https://doi.org/10.1016/B978-0-12-398512-5.00005-0. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9780123985125000050.

[72] J. Kaplan et al., *Scaling laws for neural language models*, 2020. arXiv: 2001.08361 [cs.LG]. [Online]. Available: https://arxiv.org/abs/2001.08361.

[73] T. K. Kim, "Understanding one-way anova using conceptual figures," *Korean Journal of Anesthesiology*, vol. 70, pp. 22–26, Jan. 2017. DOI: 10.4097/kjae.2017.70.1.22.

[74] L. St»hle and S. Wold, "Analysis of variance (anova)," *Chemometrics and Intelligent Laboratory Systems*, vol. 6, no. 4, pp. 259–272, 1989, ISSN: 0169-7439. DOI: https://doi.org/10.1016/0169-7439(89)80095-4. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0169743989800954.

A

Appendix 1

The appendix presents additional information about the different datasets. This includes additional information about the physicochemical properties of each dataset created along with plots highlighting the occurrence of different protein targets, E3 ligases, and cell lines.

A.1 Exact Scaffold Filtered Dataset

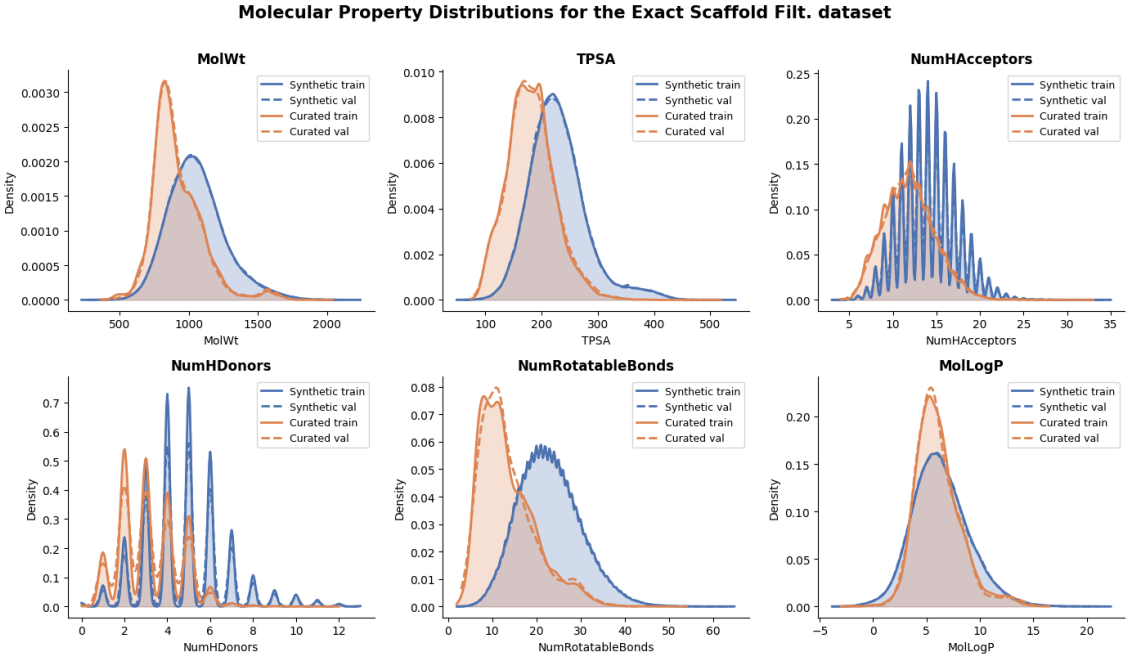


Figure A.1: The figure shows the distribution of molecular weight, tPSA, logP, hydrogen bond acceptors, hydrogen bond donors, and the number of rotatable bonds for the "Exact Scaffold Filtered" dataset.

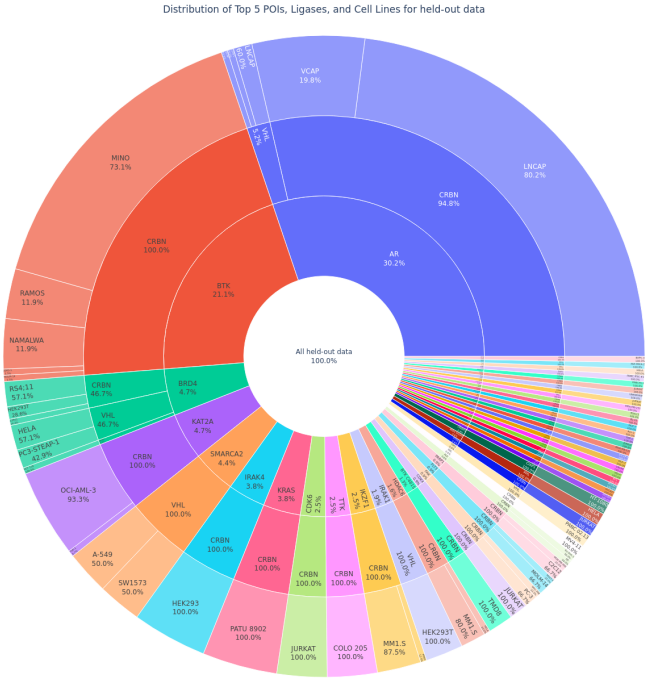


Figure A.2: Plot over the distribution of POI, E3 and Cell line in the held-out dataset. The inner-most circle shows the distribution of POI across the dataset. The second layer shows the distribution of E3's for the parent POI. Finally, the outermost layer shows the distribution of Cell lines for the parent E3 and POI.

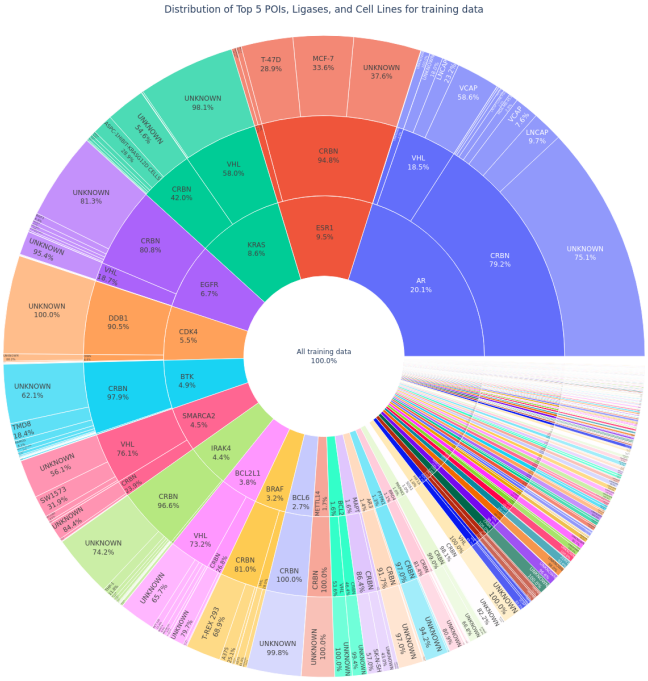


Figure A.3: Plot over the distribution of POI, E3 and Cell line in the curated training dataset. The inner-most circle shows the distribution of POI across the dataset. The second layer shows the distribution of E3's for the parent POI. Finally, the outermost layer shows the distribution of Cell lines for the parent E3 and POI.

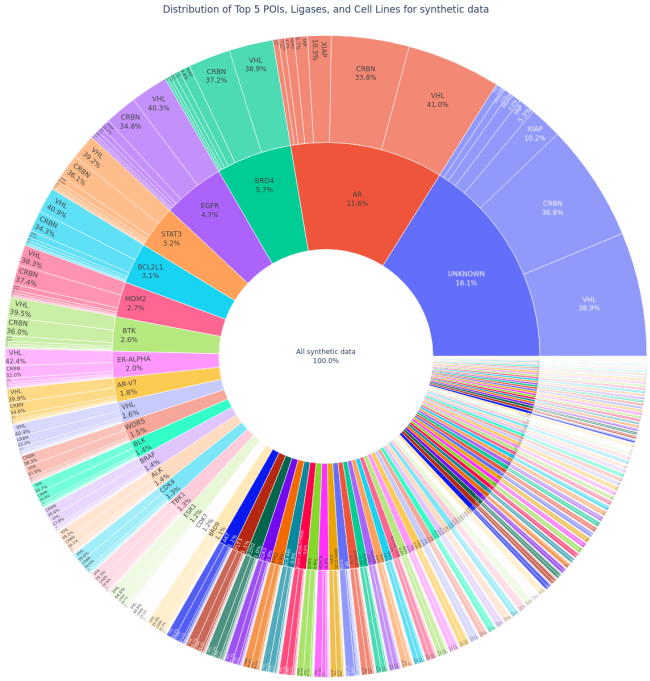


Figure A.4: Plot over the distribution of POI and E3 in the synthetic dataset. The POI and E3 are inferred based on the similarity to known POIs and E3s in the training datasets. The inner-most circle shows the distribution of POI across the dataset. The second layer shows the distribution of E3's for the parent POI.

A.2 Similarity-Threshold Dataset

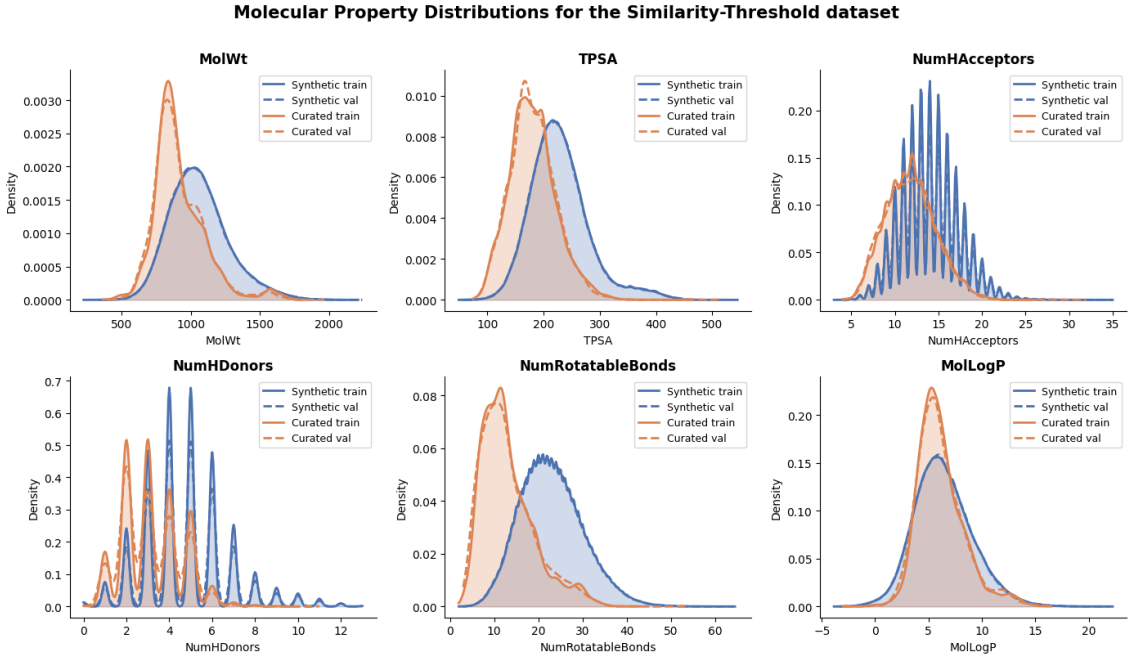


Figure A.5: The figure shows the distribution of molecular weight, tPSA, logP, hydrogen bond acceptors, hydrogen bond donors, and the number of rotatable bonds for the "Similarity-Threshold" Dataset.

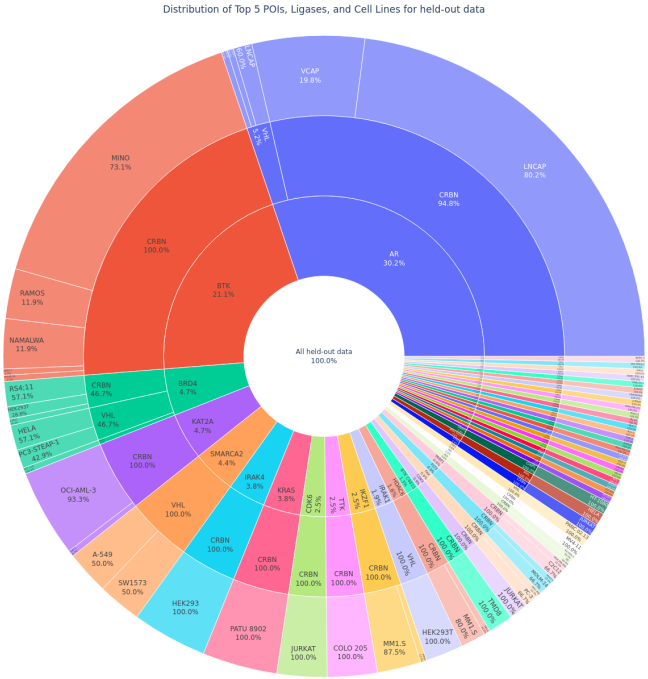


Figure A.6: Plot over the distribution of POI, E3 and Cell line in the held-out dataset. The inner-most circle shows the distribution of POI across the dataset. The second layer shows the distribution of E3's for the parent POI. Finally, the outermost layer shows the distribution of Cell lines for the parent E3 and POI.

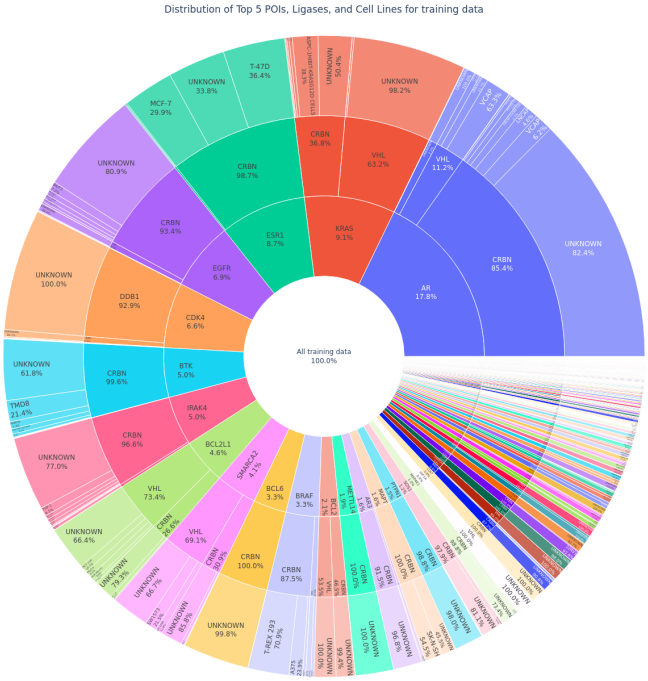


Figure A.7: Plot over the distribution of POI, E3 and Cell line in the curated training dataset. The inner-most circle shows the distribution of POI across the dataset. The second layer shows the distribution of E3's for the parent POI. Finally, the outermost layer shows the distribution of Cell lines for the parent E3 and POI.

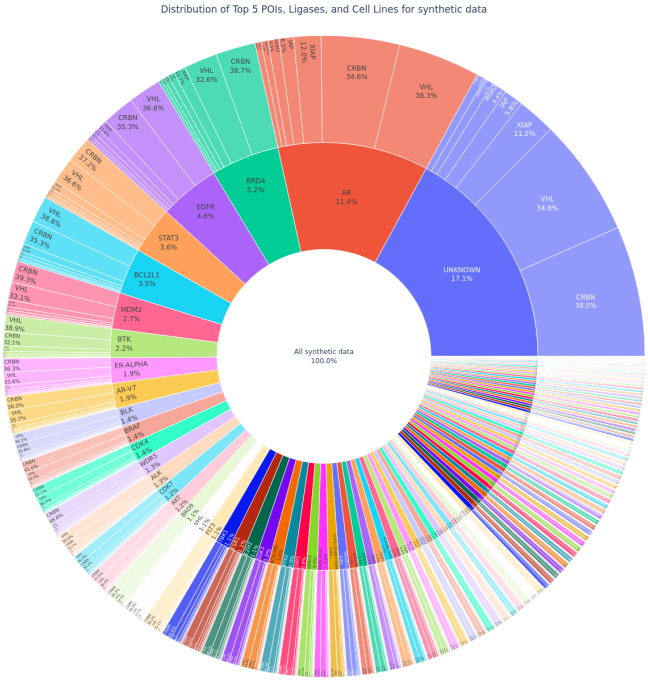


Figure A.8: Plot over the distribution of POI and E3 in the synthetic dataset. The POI and E3 are inferred based on the similarity to known POIs and E3s in the training datasets. The inner-most circle shows the distribution of POI across the dataset. The second layer shows the distribution of E3's for the parent POI.

A.3 No AR Dataset

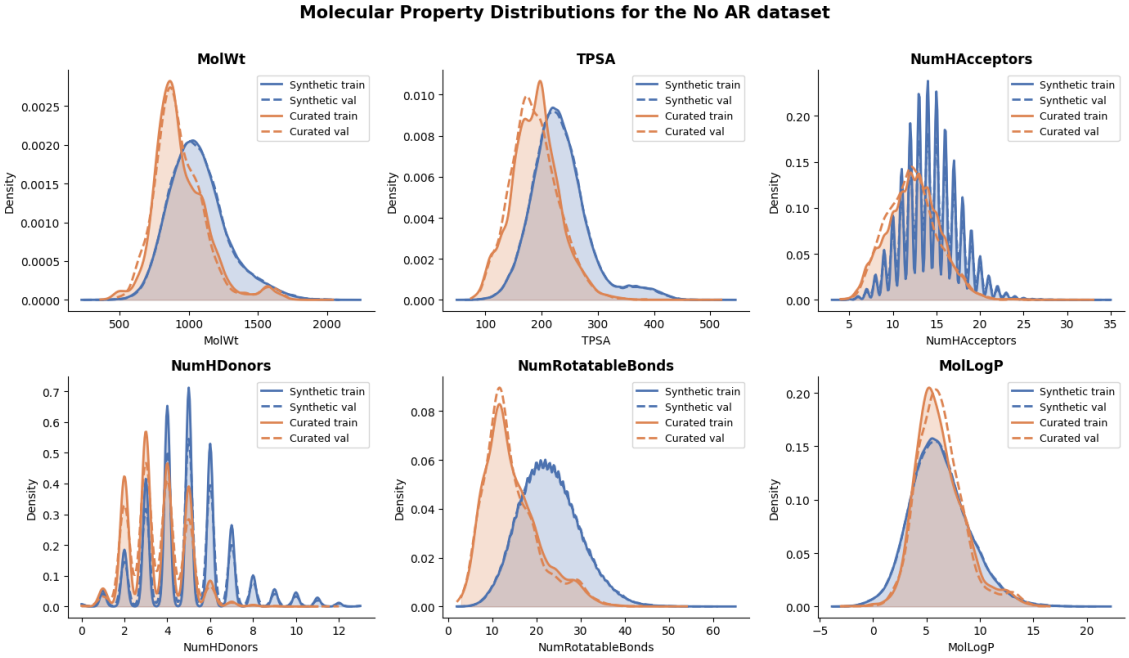


Figure A.9: The figure shows the distribution of molecular weight, tPSA, logP, hydrogen bond acceptors, hydrogen bond donors, and the number of rotatable bonds for the "No AR" Dataset.

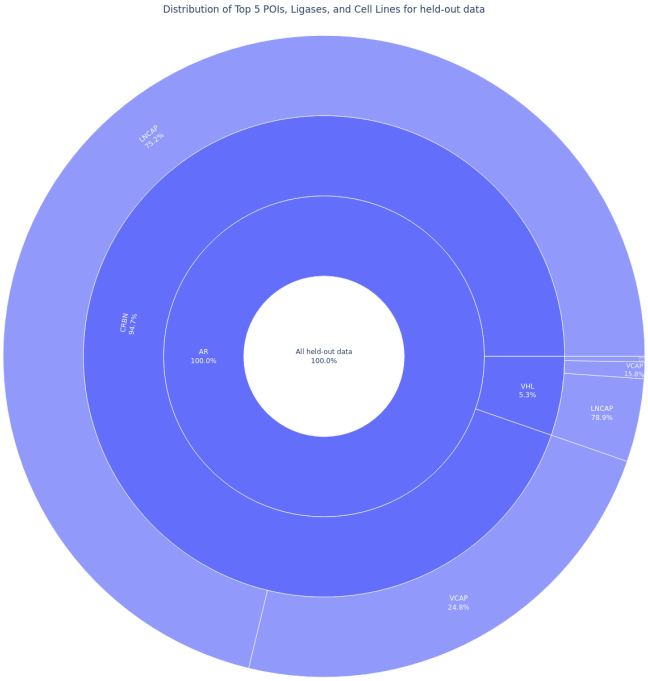


Figure A.10: Plot over the distribution of POI, E3 and Cell line in the held-out dataset. The inner-most circle shows the distribution of POI across the dataset. The second layer shows the distribution of E3's for the parent POI. Finally, the outermost layer shows the distribution of Cell lines for the parent E3 and POI.

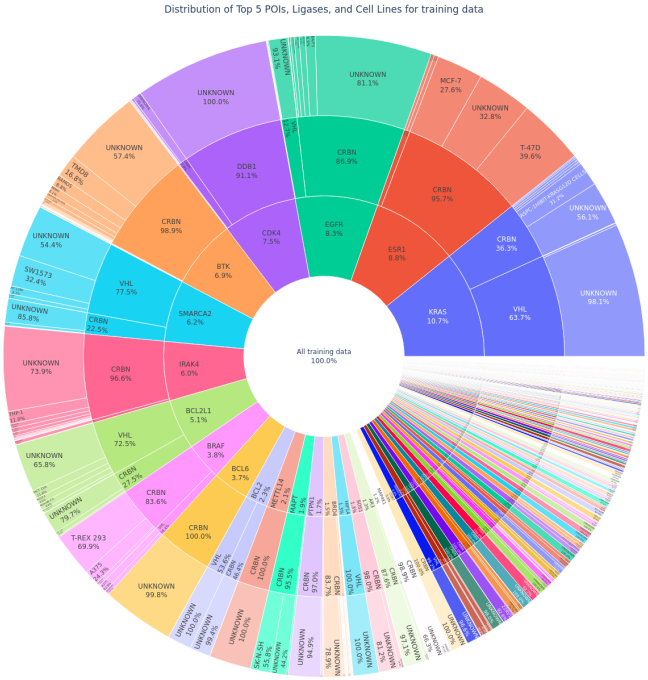


Figure A.11: Plot over the distribution of POI, E3 and Cell line in the curated training dataset. The inner-most circle shows the distribution of POI across the dataset. The second layer shows the distribution of E3's for the parent POI. Finally, the outermost layer shows the distribution of Cell lines for the parent E3 and POI.

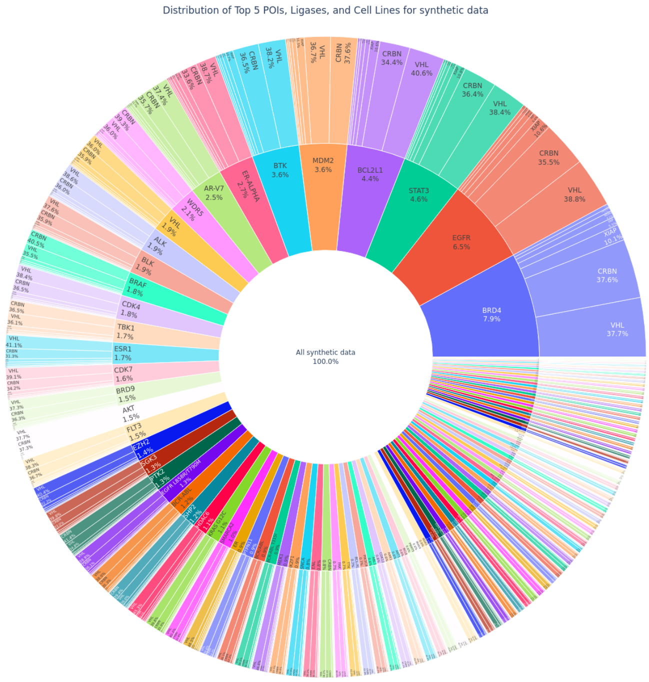


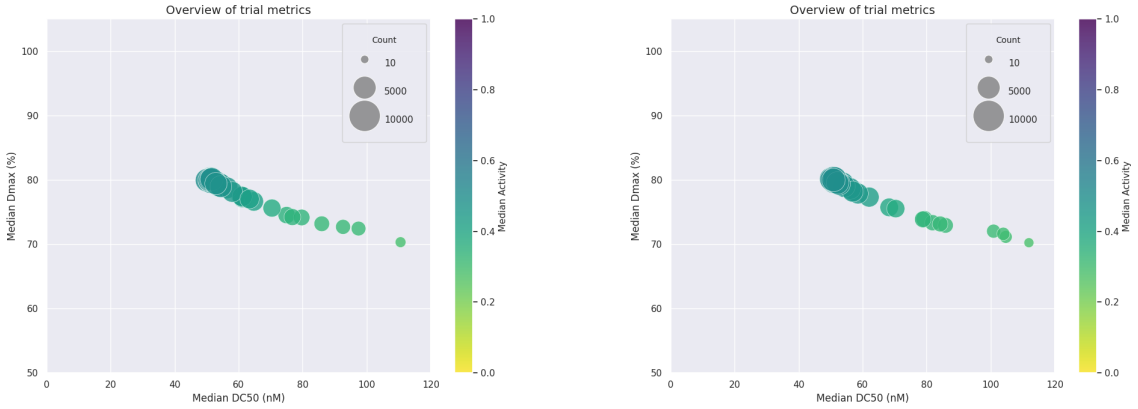
Figure A.12: Plot over the distribution of POI and E3 in the synthetic dataset. The POI and E3 are inferred based on the similarity to known POIs and E3s in the training datasets. The inner-most circle shows the distribution of POI across the dataset. The second layer shows the distribution of E3's for the parent POI.

B

Appendix 2

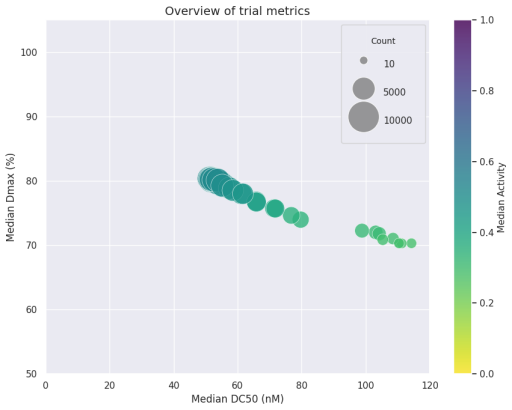
B.1 REINVENT LSTM Optuna Trials

B.1.1 Synthetic Transfer Learning



(a)

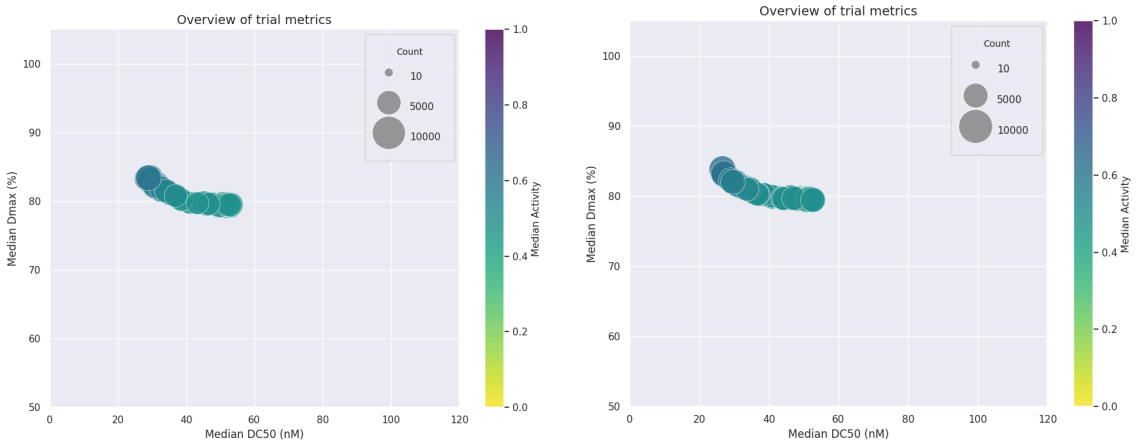
(b)



(c)

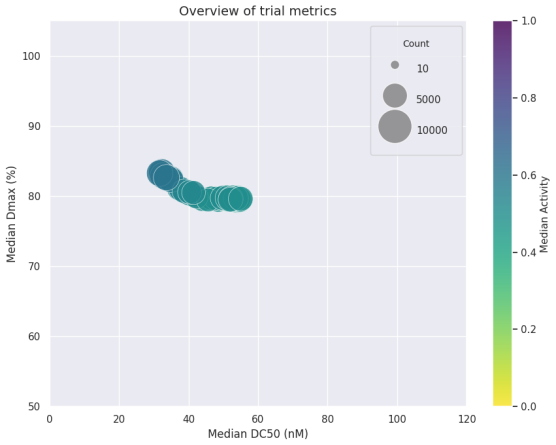
Figure B.1: This figure shows the median Activity, DC50, and Dmax for different configurations of hyperparameters using Optuna on the synthetic transfer learning. The metrics are calculated on SMILES that surpass the activity threshold of > 0.5 predicted by the binary predictor. The datasets are: a) the Exact Scaffold Filt. b) the Sim.-Threshold, c) the No AR

B.1.2 Curated Transfer Learning



(a)

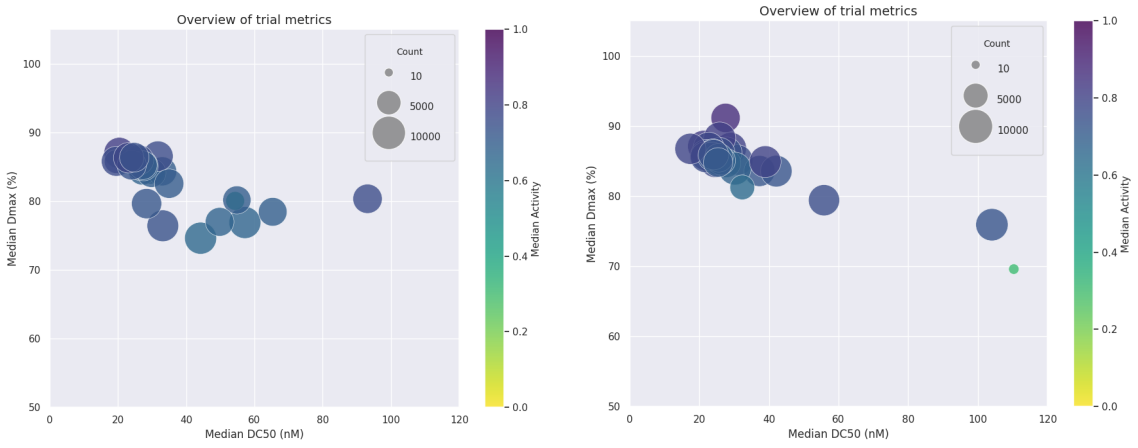
(b)



(c)

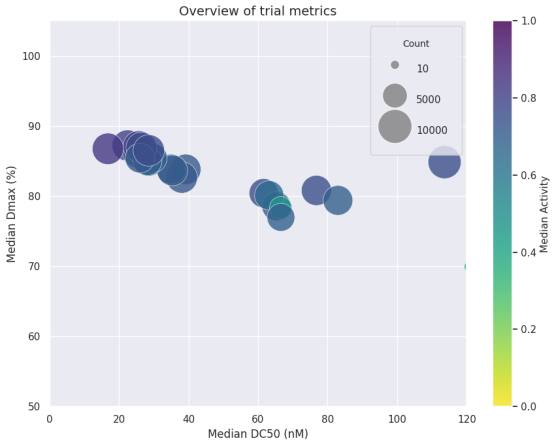
Figure B.2: This figure shows the median Activity, DC50 and Dmax for different configurations of hyperparameters using Optuna on the curated transfer learning. The metrics are calculated on SMILES that surpass the activity threshold of > 0.5 predicted by the binary predictor. The datasets are: a) the Exact Scaffold Filt. b) the Sim.-Threshold, c) the No AR

B.1.3 Reinforcement Learning



(a)

(b)

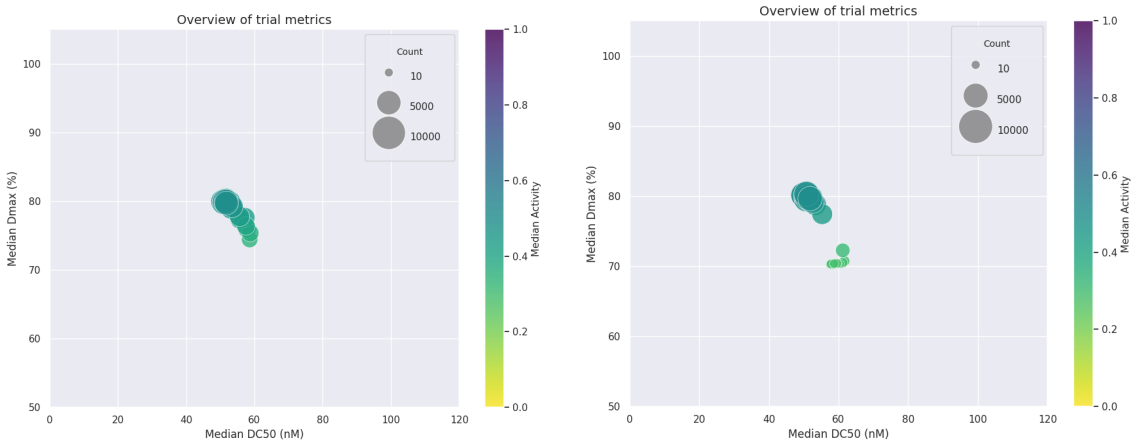


(c)

Figure B.3: This figure shows the median Activity, DC50, and Dmax for different configurations of hyperparameters using Optuna on reinforcement learning. The metrics are calculated on SMILES that surpass the activity threshold of > 0.5 predicted by the binary predictor. The datasets are: a) the Exact Scaffold Filt. b) the Sim.-Threshold, c) the No AR

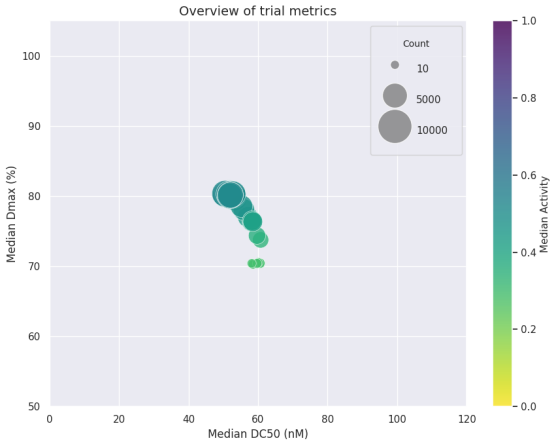
B.2 Transformer Optuna Trials

B.2.1 Synthetic Transfer Learning



(a)

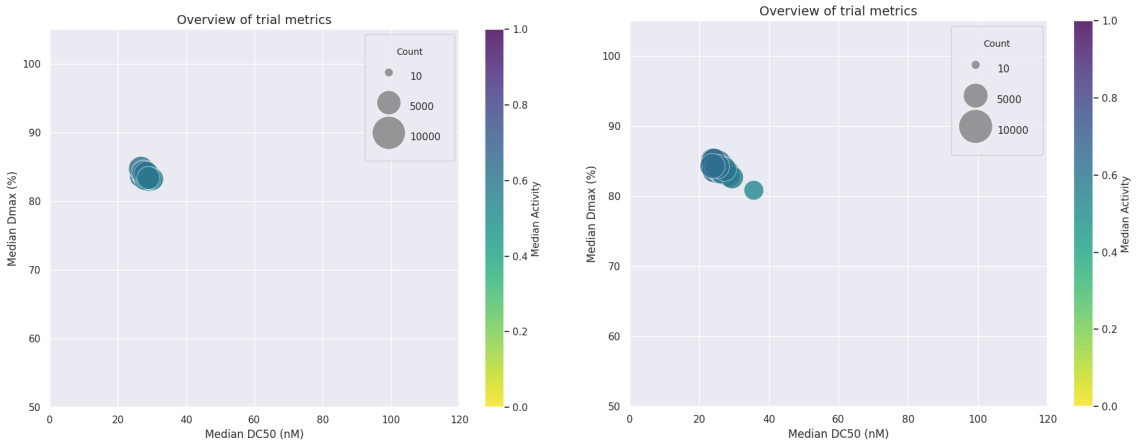
(b)



(c)

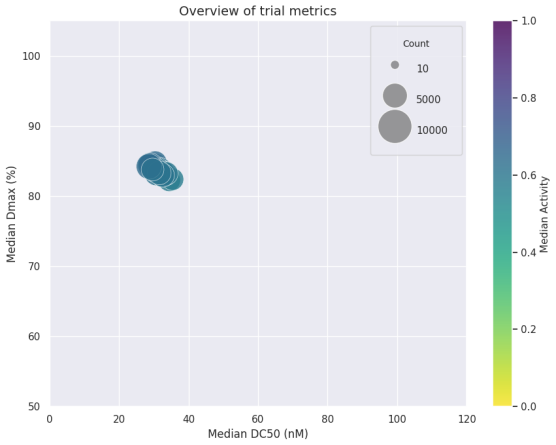
Figure B.4: This figure shows the median Activity, DC50 and Dmax for different configurations of hyperparameters using Optuna on reinforcement learning. The metrics are calculated on SMILES that surpass the activity threshold of > 0.5 predicted by the binary predictor. The datasets corresponds to: a) the Exact Scaffold Filt. b) the Sim.-Threshold, c) the No AR

B.2.2 Curated Transfer Learning



(a)

(b)



(c)

Figure B.5: This figure shows the median Activity, DC50, and Dmax for different configurations of hyperparameters using Optuna on reinforcement learning. The metrics are calculated on SMILES that surpass the activity threshold of > 0.5 predicted by the binary predictor. The datasets are: a) the Exact Scaffold Filt. b) the Sim.-Threshold, c) the No AR

B.2.3 Reinforcement Learning

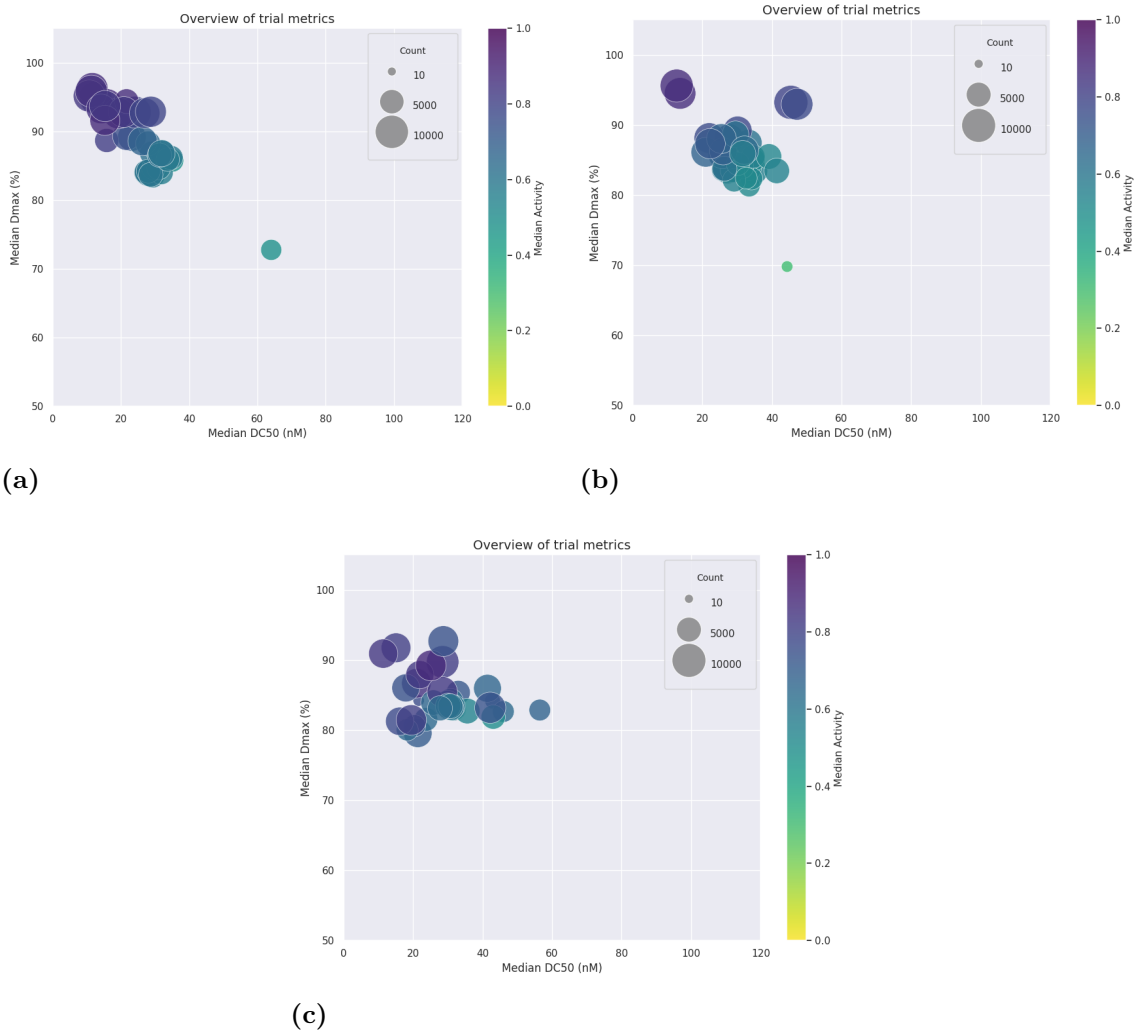


Figure B.6: This figure shows the median Activity, DC50, and Dmax for different configurations of hyperparameters using Optuna on reinforcement learning. The metrics are calculated on SMILES that surpass the activity threshold of > 0.5 predicted by the binary predictor. The datasets are: a) the Exact Scaffold Filt. b) the Sim.-Threshold, c) the No AR

B.3 Pareto Front Compounds

In this section of the appendix, the compounds that constitute each of the Pareto fronts are visualized for each model.

B.3.1 LSTM Pareto Front Compounds

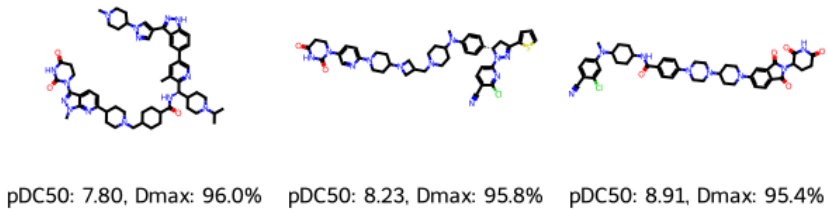


Figure B.7: Pareto front compounds for the LSTM trained on the exact scaffold filtered dataset.

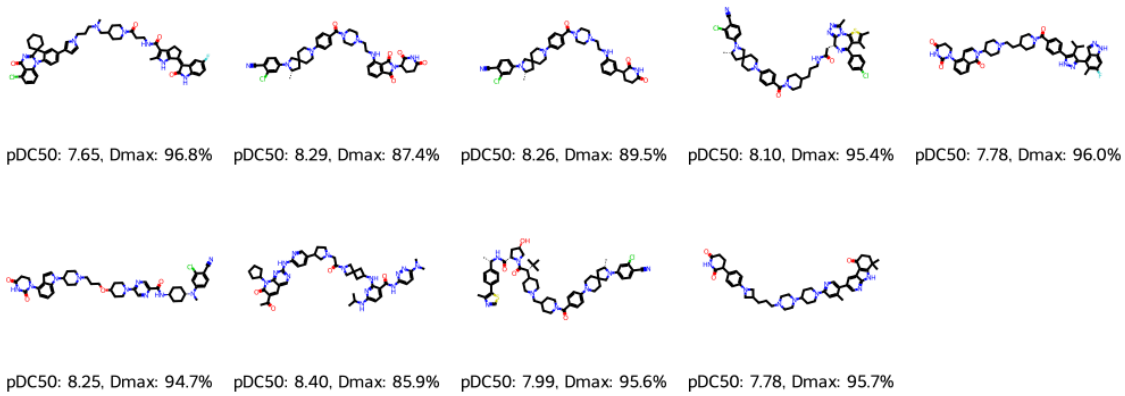


Figure B.8: Pareto front compounds for the LSTM trained on the similarity-threshold dataset.

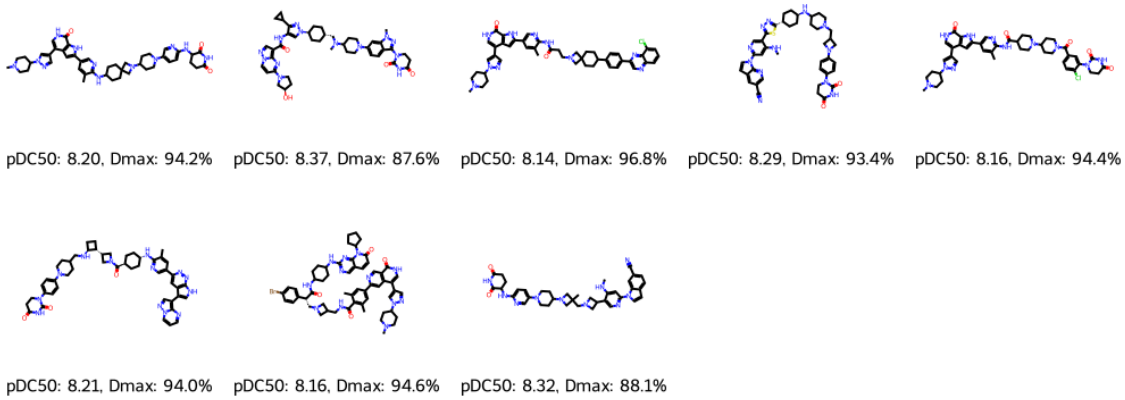


Figure B.9: Pareto front compounds for the LSTM trained on the no AR dataset.

B.3.2 Transformer Pareto Front Compounds

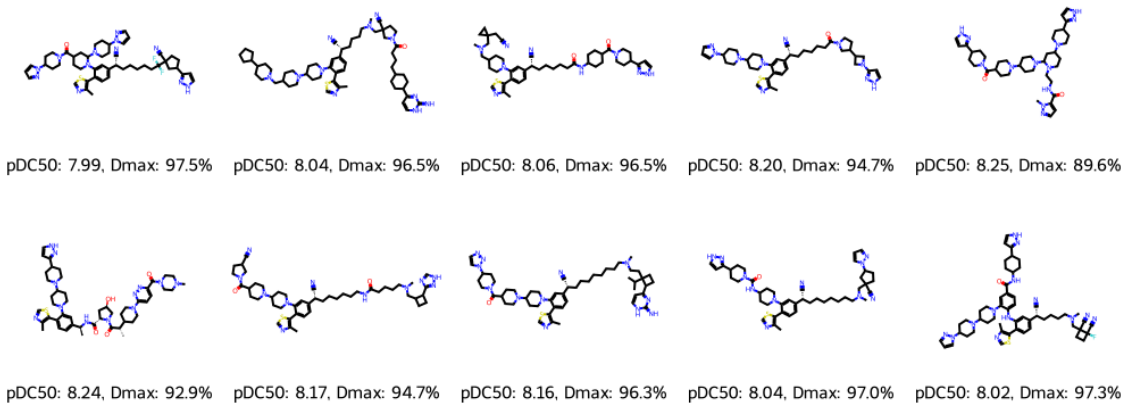


Figure B.10: Pareto front compounds for the Transformer trained on the exact scaffold filtered dataset.

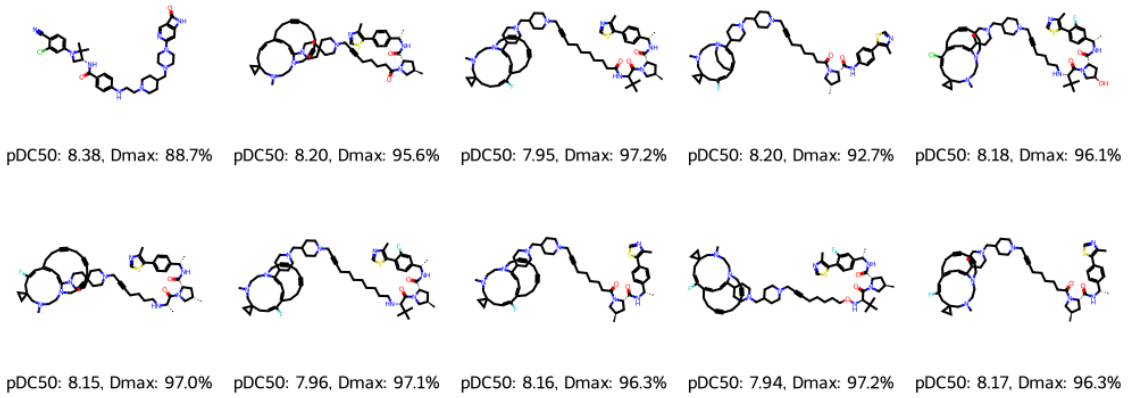


Figure B.11: Pareto front compounds for the Transformer trained on the similarity-threshold dataset.

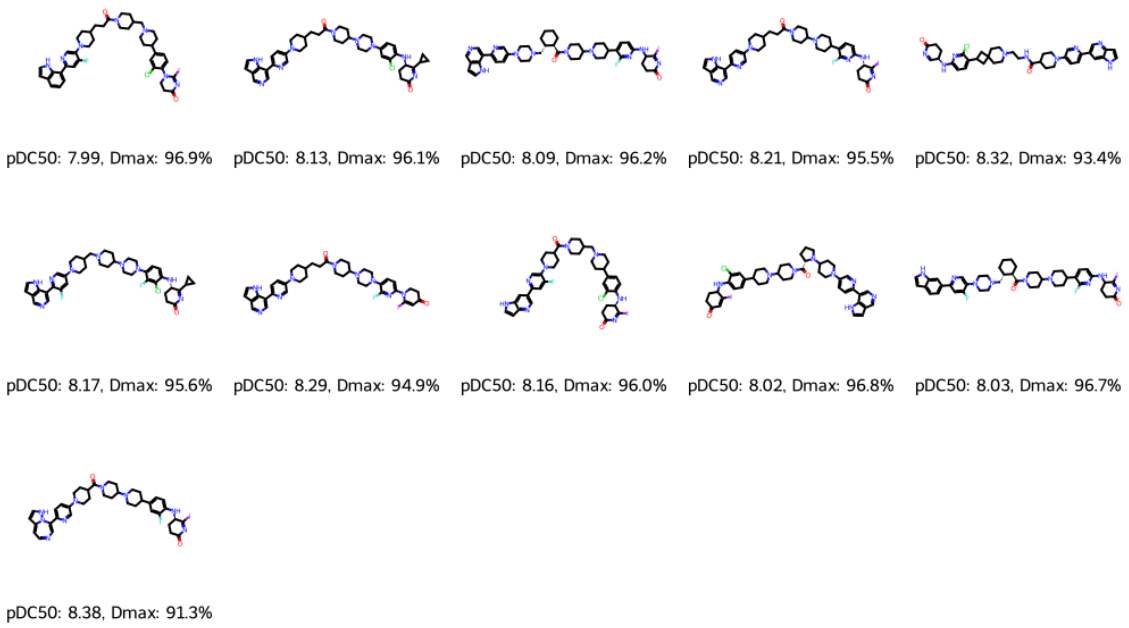


Figure B.12: Pareto front compounds for the Transformer trained on the no AR dataset.

B.4 Similarity to Held-out for Best Generated PROTACs

B.4.1 REINVENT LSTM PROTACs

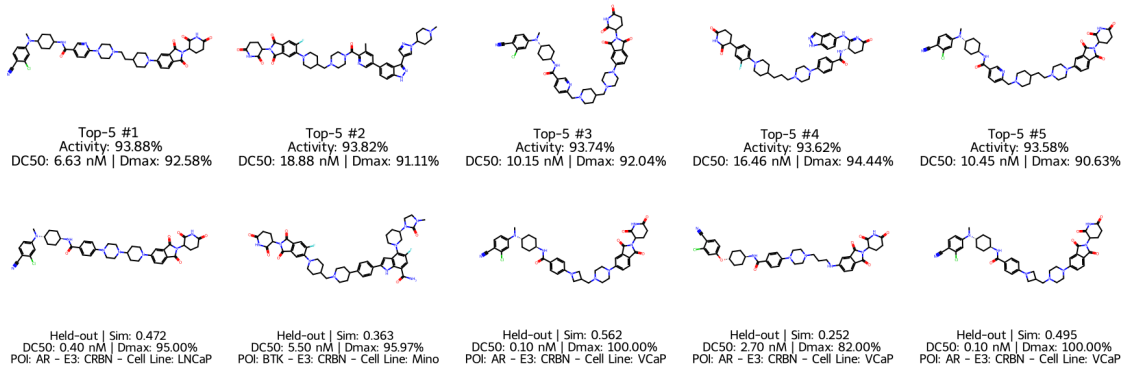


Figure B.13: Top-5 generated PROTACs with the highest activity score from the REINVENT LSTM model trained on the exact scaffold filtered dataset (top), and the PROTACs they are most similar to, respectively, from the held-out dataset (bottom). The figure also displays the POI, E3, and cell line for which the PROTACs from the held-out are active.

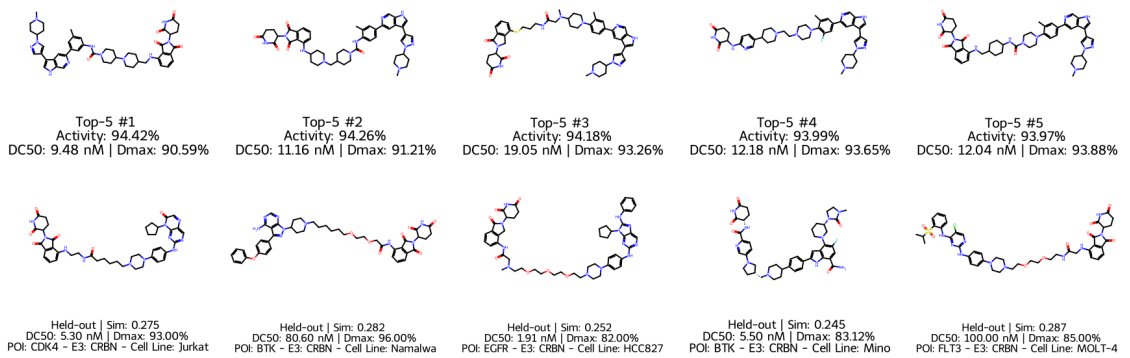


Figure B.14: Top-5 generated PROTACs with the highest activity score from the REINVENT LSTM model trained on the similarity-threshold dataset (top), and the PROTACs they are most similar to, respectively, from the held-out dataset (bottom). The figure also displays the POI, E3, and cell line for which the PROTACs from the held-out are active.

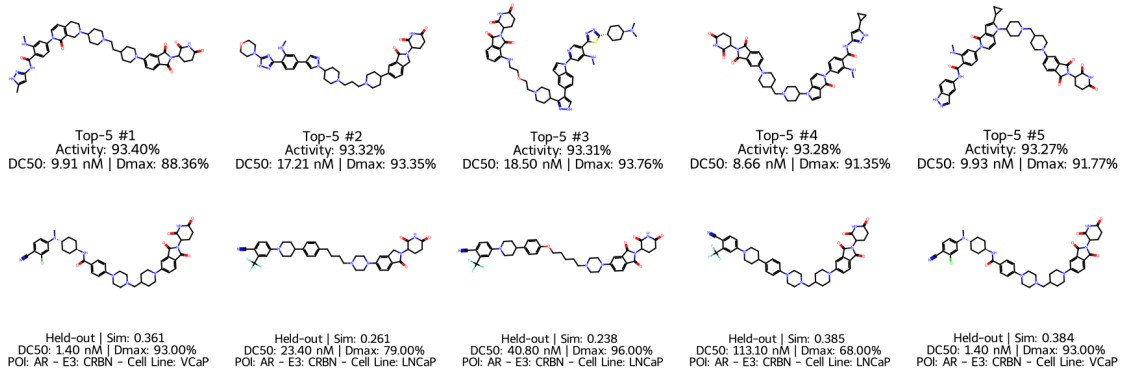


Figure B.15: Top-5 generated PROTACs with the highest activity score from the REINVENT LSTM model trained on the no AR dataset (top), and the PROTACs they are most similar to, respectively, from the held-out dataset (bottom). The figure also displays the POI, E3, and cell line for which the PROTACs from the held-out are active.

B.4.2 Transformer PROTACs

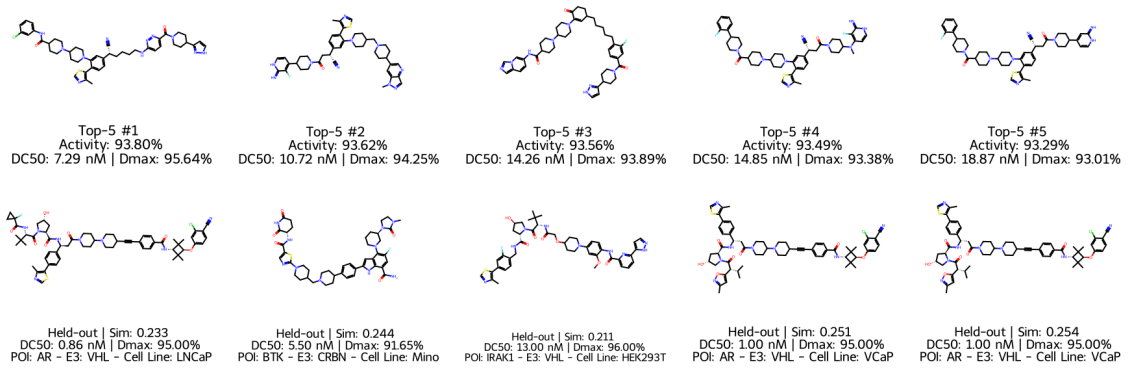


Figure B.16: Top-5 generated PROTACs with the highest activity score from the Transformer model trained on the exact scaffold filtered dataset (top), and the PROTACs they are most similar to, respectively, from the held-out dataset (bottom). The figure also displays the POI, E3, and cell line for which the PROTACs from the held-out are active.

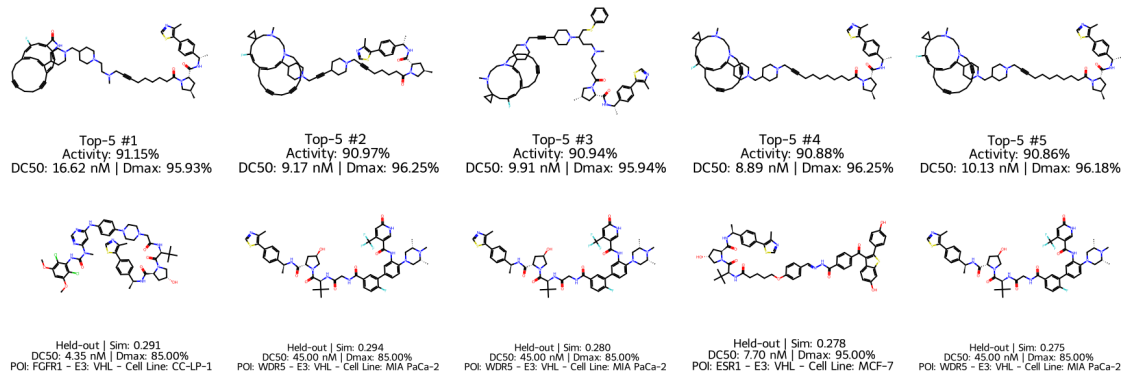


Figure B.17: Top-5 generated PROTACs with the highest activity score from the Transformer model trained on the similarity-threshold dataset (top), and the PROTACs they are most similar to, respectively, from the held-out dataset (bottom). The figure also displays the POI, E3, and cell line for which the PROTACs from the held-out are active.

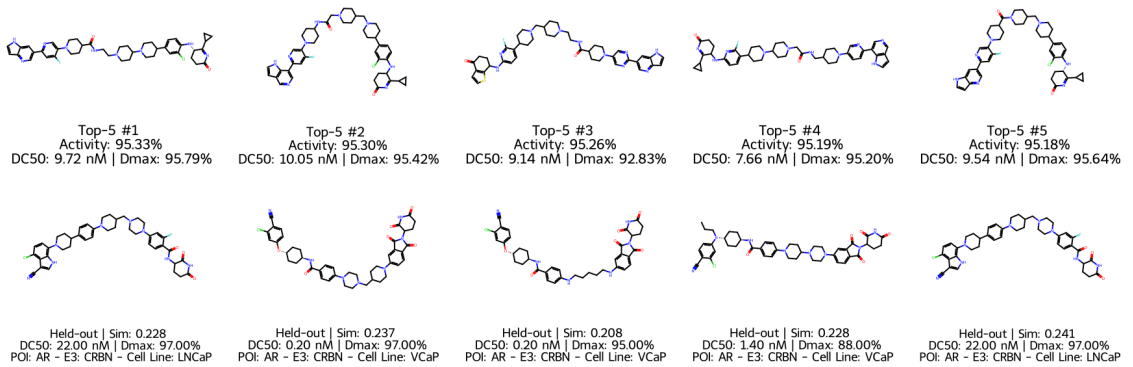


Figure B.18: Top-5 generated PROTACs with the highest activity score from the Transformer model trained on the no AR dataset (top), and the PROTACs they are most similar to, respectively, from the held-out dataset (bottom). The figure also displays the POI, E3, and cell line for which the PROTACs from the held-out are active.

B.4.3 Evaluation metrics

B.4.3.1 Identical runs table

Table B.1 highlights these evaluation metrics across TL and RL for an LSTM backbone model and a Transformer backbone model along with the different datasets. Each configuration presents the mean and the standard deviation across the 5 models trained with identical configurations. For each metric, the best value is marked in bold, and the best value for each step (TL: Synthetic, TL: Curated, and RL) is highlighted in blue, orange, and green, respectively.

Dataset	Training step	Backbone model	V (%) ↑	U (%) ↑	N (%) ↑	Diversity ↑	SA Avg ↓
Full	TL: Synthetic	REINVENT (LSTM)	$96.6_{\pm 0.114}$	$99.9_{\pm 0.015}$	$93.8_{\pm 0.319}$	$0.792_{\pm 0.000}$	$5.07_{\pm 0.005}$
		Transformer	$96.4_{\pm 1.82}$	$99.9_{\pm 0.040}$	$85.6_{\pm 2.86}$	$0.787_{\pm 0.000}$	$5.10_{\pm 0.008}$
	TL: Curated	REINVENT (LSTM)	$88.5_{\pm 0.295}$	$95.4_{\pm 0.121}$	$89.0_{\pm 0.253}$	$0.811_{\pm 0.000}$	**4.01**$_{\pm 0.010}$
		Transformer	$80.7_{\pm 1.48}$	$91.6_{\pm 0.738}$	$84.6._{\pm 2.68}$	$0.806_{\pm 0.000}$	$4.75_{\pm 0.012}$
	RL	REINVENT (LSTM)	$97.2_{\pm 1.13}$	$93.8_{\pm 1.54}$	$99.7_{\pm 0.210}$	$0.775_{\pm 0.010}$	$4.42_{\pm 0.065}$
		Transformer	$93.9_{\pm 1.51}$	$96.2_{\pm 2.67}$	**100**$_{\pm 0.000}$	$0.699_{\pm 0.045}$	$5.40_{\pm 0.521}$
Similarity-threshold	TL: Synthetic	REINVENT (LSTM)	$96.7_{\pm 0.153}$	**100**$_{\pm 0.017}$	$94.3_{\pm 0.293}$	$0.794_{\pm 0.000}$	$5.08_{\pm 0.009}$
		Transformer	$96.6_{\pm 1.13}$	$99.9_{\pm 0.005}$	$85.9_{\pm 1.64}$	$0.789_{\pm 0.001}$	$5.11_{\pm 0.004}$
	TL: Curated	REINVENT (LSTM)	$86.0_{\pm 0.388}$	$96.1_{\pm 0.203}$	$92.6_{\pm 0.197}$	**0.818**$_{\pm 0.001}$	$4.70_{\pm 0.006}$
		Transformer	$75.9_{\pm 0.384}$	$92.0_{\pm 0.049}$	$86.3_{\pm 0.551}$	$0.811_{\pm 0.001}$	$4.75_{\pm 0.014}$
	RL	REINVENT (LSTM)	**97.8**$_{\pm 0.926}$	$89.4_{\pm 7.288}$	$99.9_{\pm 0.92}$	$0.703_{\pm 0.051}$	$4.49_{\pm 0.159}$
		Transformer	$93.9_{\pm 1.57}$	$90.1_{\pm 2.74}$	**100**$_{\pm 0.031}$	$0.679_{\pm 0.052}$	$5.33_{\pm 0.975}$
no AR target POI	TL: Synthetic	REINVENT (LSTM)	$95.7_{\pm 0.258}$	$99.9_{\pm 0.032}$	$95.1_{\pm 0.328}$	$0.794\_\pm 0.001$	$5.06_{\pm 0.009}$
		Transformer	$95.8_{\pm 1.53}$	$99.9_{\pm 0.006}$	$85.1_{\pm 2.46}$	$0.786_{\pm 0.001}$	$5.12_{\pm 0.009}$
	TL: Curated	REINVENT (LSTM)	$86.9_{\pm 0.290}$	$95.8_{\pm 0.171}$	$91.0_{\pm 0.291}$	$0.815_{\pm 0.001}$	$4.77_{\pm 0.008}$
		Transformer	$75.0_{\pm 1.51}$	$90.6_{\pm 0.764}$	$85.9_{\pm 2.23}$	$0.806_{\pm 0.000}$	$4.78_{\pm 0.005}$
	RL	REINVENT (LSTM)	$97.0_{\pm 0.640}$	$92.5_{\pm 5.40}$	$99.7_{\pm 0.177}$	$0.766_{\pm 0.025}$	$4.62_{\pm 0.144}$
		Transformer	$87.07_{\pm 6.10}$	$90.7_{\pm 2.79}$	**100**$_{\pm 0.020}$	$0.717_{\pm 0.033}$	$4.47_{\pm 0.267}$

Table B.1: Summary of evaluation metrics across different models, training steps, and datasets. Here, the mean of the mean of the 5 identical runs is presented ± 1 standard deviation. The table highlights the best-performing model for each training step through its colors. A value marked with blue corresponds to the overall best model for the synthetic TL step; likewise, orange indicates the best model on curated TL, and green indicates the best RL model. Additionally, a value marked in bold represents the best performance per evaluation metric. The columns V, U, and N represent the validity, the uniqueness, and the novelty, respectively.

B.4.3.2 Physicochemical Properties of Identical Runs

Dataset	Training step	Backbone model	MW	HBA	HBD	RotB	tPSA	logP
Full	TL: Synthetic	REINVENT (LSTM)	$1040.1_{\pm 1.581}$	$13.83_{\pm 0.018}$	$4.725_{\pm 0.012}$	$22.58_{\pm 0.047}$	$255.6_{\pm 0.201}$	$6.222_{\pm 0.029}$
		Transformer	$1049.9_{\pm 3.165}$	$13.98_{\pm 0.066}$	$4.740_{\pm 0.007}$	$22.80_{\pm 0.0257}$	$227.5_{\pm 0.665}$	$6.295_{\pm 0.015}$
	TL: Curated	REINVENT (LSTM)	$875.8_{\pm 1.240}$	$11.32_{\pm 0.026}$	$3.078_{\pm 0.012}$	$13.04_{\pm 0.048}$	$174.2_{\pm 0.361}$	$5.789_{\pm 0.011}$
		Transformer	$888.6_{\pm 1.738}$	$11.44_{\pm 0.095}$	$3.130_{\pm 0.017}$	$12.96_{\pm 0.016}$	$175.6_{\pm 1.418}$	$5.981_{\pm 0.072}$
	RL	REINVENT (LSTM)	$883.4_{\pm 25.81}$	$11.29_{\pm 0.800}$	$2.979_{\pm 0.358}$	$13.82_{\pm 1.117}$	$170.3_{\pm 5.830}$	$5.871_{\pm 0.272}$
		Transformer	$948.9_{\pm 85.75}$	$10.82_{\pm 1.157}$	$3.545_{\pm 1.017}$	$18.31_{\pm 4.282}$	$145.6_{\pm 19.46}$	$8.943_{\pm 0.693}$
Similarity-threshold	TL: Synthetic	REINVENT (LSTM)	$1042.8_{\pm 2.985}$	$13.76_{\pm 0.043}$	$4.729_{\pm 0.029}$	$22.63_{\pm 0.087}$	$225.6_{\pm 1.115}$	$6.296_{\pm 0.031}$
		Transformer	$1052.2_{\pm 2.220}$	$13.91_{\pm 0.033}$	$4.718_{\pm 0.002}$	$22.82_{\pm 0.068}$	$227.4_{\pm 0.305}$	$6.366_{\pm 0.037}$
	TL: Curated	REINVENT (LSTM)	$867.4_{\pm 1.106}$	$11.14_{\pm 0.015}$	$3.042_{\pm 0.008}$	$12.57_{\pm 0.045}$	$172.4_{\pm 0.226}$	$5.769_{\pm 0.018}$
		Transformer	$876.5_{\pm 0.370}$	$11.23_{\pm 0.033}$	$3.102_{\pm 0.016}$	$12.37_{\pm 0.017}$	$173.1_{\pm 0.605}$	$5.895_{\pm 0.037}$
	RL	REINVENT (LSTM)	$884.2_{\pm 32.05}$	$10.21_{\pm 1.740}$	$3.159_{\pm 0.415}$	$14.50_{\pm 2.635}$	$163.2_{\pm 15.99}$	$6.112_{\pm 0.755}$
		Transformer	$977.7_{\pm 153.0}$	$11.15_{\pm 2.407}$	$3.600_{\pm 1.517}$	$16.96_{\pm 3.852}$	$157.0_{\pm 50.27}$	$7.875_{\pm 1.074}$
no AR target POI	TL: Synthetic	REINVENT (LSTM)	$1041.6_{\pm 1.381}$	$13.99_{\pm 0.024}$	$4.858_{\pm 0.022}$	$22.50_{\pm 0.053}$	$228.2_{\pm 0.677}$	$6.054_{\pm 0.021}$
		Transformer	$1062.4_{\pm 4.317}$	$14.28_{\pm 0.067}$	$4.876_{\pm 0.011}$	$22.81_{\pm 0.048}$	$232.0_{\pm 0.594}$	$6.216_{\pm 0.053}$
	TL: Curated	REINVENT (LSTM)	$890.7_{\pm 3.025}$	$11.75_{\pm 0.053}$	$3.385_{\pm 0.019}$	$13.96_{\pm 0.095}$	$181.1_{\pm 0.871}$	$5.837_{\pm 0.027}$
		Transformer	907.6 ± 1.547	$12.05_{\pm 0.063}$	$3.483_{\pm 0.030}$	$14.31_{\pm 0.128}$	$184.8_{\pm 0.769}$	$5.961_{\pm 0.002}$
	RL	REINVENT (LSTM)	$895.3_{\pm 37.22}$	$11.25_{\pm 0.753}$	$3.088_{\pm 0.226}$	$13.27_{\pm 1.378}$	$173.0_{\pm 9.249}$	$6.010_{\pm 0.626}$
		Transformer	$827.5_{\pm 62.71}$	$8.288_{\pm 1.544}$	$3.404_{\pm 0.733}$	$12.81_{\pm 1.870}$	$120.9_{\pm 27.71}$	$7.744_{\pm 0.284}$

Table B.2: Summary of physicochemical properties across different models, training steps, and datasets. The table highlights the best-performing model for each training step through its colors. A value marked with blue corresponds to the overall best model for the synthetic TL step; likewise, orange indicates the best model on curated TL, and green indicates the best RL model. Additionally, a value marked in bold represents the best performance per evaluation metric. MW = molecular weight, HBA = hydrogen bond acceptors, HBD = hydrogen bond donors, RotB = rotatable bonds, tPSA = topological polar surface area, logP = octanol-water partition coefficient.

B.4.3.3 Evaluation metrics best runs

Dataset	Training step	Backbone model	V (%) ↑	U (%) ↑	N (%) ↑	Diversity ↑	SA Avg ↓
Exact Scaffold Filt.	TL: Synthetic	REINVENT (LSTM)	$96.7_{\pm 0.423}$	$100_{\pm 0.000}$	$93.9_{\pm 0.631}$	$0.791_{\pm 0.001}$	$5.07_{\pm 0.021}$
		Transformer	$92.7_{\pm 0.935}$	$100_{\pm 0.034}$	$91.3_{\pm 0.707}$	$0.789_{\pm 0.002}$	$5.08_{\pm 0.023}$
	TL: Curated	REINVENT (LSTM)	$89.1_{\pm 0.944}$	$99.3_{\pm 0.329}$	$86.7_{\pm 0.930}$	$0.815_{\pm 0.001}$	$4.696_{\pm 0.012}$
		Transformer	$83.7_{\pm 0.918}$	$98.1_{\pm 0.442}$	$75.4_{\pm 1.21}$	$0.801_{\pm 0.002}$	$4.77_{\pm 0.041}$
	RL	REINVENT (LSTM)	$97.9_{\pm 0.383}$	$98.9_{\pm 0.347}$	$99.8_{\pm 0.186}$	$0.763_{\pm 0.002}$	$4.45_{\pm 0.019}$
		Transformer	$95.6_{\pm 0.56}$	$99.1_{\pm 0.283}$	$100_{\pm 0.000}$	$0.724_{\pm 0.002}$	$4.53_{\pm 0.019}$
Similarity-threshold	TL: Synthetic	REINVENT (LSTM)	$96.7_{\pm 0.631}$	$100_{\pm 0.033}$	$94.5_{\pm 0.592}$	$0.794_{\pm 0.001}$	$5.07_{\pm 0.025}$
		Transformer	$94.3_{\pm 0.70}$	$100_{\pm 0.03}$	$89.2_{\pm 0.741}$	$0.791_{\pm 0.001}$	$5.10_{\pm 0.025}$
	TL: Curated	REINVENT (LSTM)	$86.4_{\pm 0.989}$	$99.3_{\pm 0.219}$	$90.4_{\pm 0.758}$	$0.818_{\pm 0.001}$	$4.699_{\pm 0.027}$
		Transformer	$76.7_{\pm 1.64}$	$98.4_{\pm 0.309}$	$84.6_{\pm 1.08}$	$0.810_{\pm 0.002}$	$4.71_{\pm 0.043}$
	RL	REINVENT (LSTM)	$97.9_{\pm 0.280}$	$99.0_{\pm 0.298}$	$100_{\pm 0.050}$	$0.769_{\pm 0.001}$	$4.42_{\pm 0.014}$
		Transformer	$92.6_{\pm 0.838}$	$98.4_{\pm 0.492}$	$\mathbf{100}_{\pm 0.000}$	$0.665_{\pm 0.002}$	$6.50_{\pm 0.043}$
No AR	TL: Synthetic	REINVENT (LSTM)	$95.7_{\pm 0.600}$	$100_{\pm 0.033}$	$94.7_{\pm 0.484}$	$0.794_{\pm 0.002}$	$5.05_{\pm 0.012}$
		Transformer	$92.7_{\pm 0.447}$	$100_{\pm 0.000}$	$90.0_{\pm 0.894}$	$0.789_{\pm 0.001}$	$5.11_{\pm 0.020}$
	TL: Curated	REINVENT (LSTM)	$87.2_{\pm 1.09}$	$99.4_{\pm 0.156}$	$88.7_{\pm 0.682}$	$0.815_{\pm 0.002}$	$4.77_{\pm 0.034}$
		Transformer	$78.1_{\pm 1.26}$	$98.2_{\pm 0.656}$	$76.9_{\pm 2.01}$	$0.807_{\pm 0.002}$	$4.77_{\pm 0.033}$
	RL	REINVENT (LSTM)	$96.0_{\pm 0.652}$	$99.4_{\pm 0.197}$	$99.8_{\pm 0.133}$	$0.780_{\pm 0.002}$	$4.80_{\pm 0.026}$
		Transformer	$84.5_{\pm 1.13}$	$98.8_{\pm 0.390}$	$100_{\pm 0.075}$	$0.800_{\pm 0.002}$	$4.13_{\pm 0.011}$

Table B.3: Summary of evaluation metrics across different models, training stages, and datasets. The table highlights the best-performing model for each training stage through its colors. A value marked with blue corresponds to the overall best model for the synthetic TL step; likewise, orange indicates the best model on curated TL, and green indicates the best RL model. Additionally, a value marked in bold represents the best performance per evaluation metric. The columns V, U, and N represent the validity, the uniqueness, and the novelty, respectively.

B.5 Transfer learning physicochemical properties

This section presents the distribution of physicochemical properties for generated PROTACs after the transfer learning stage. The results are presented across datasets and for both the REINVENT LSTM and the Transformer.

B.5.1 Exact Scaffold Filtered dataset

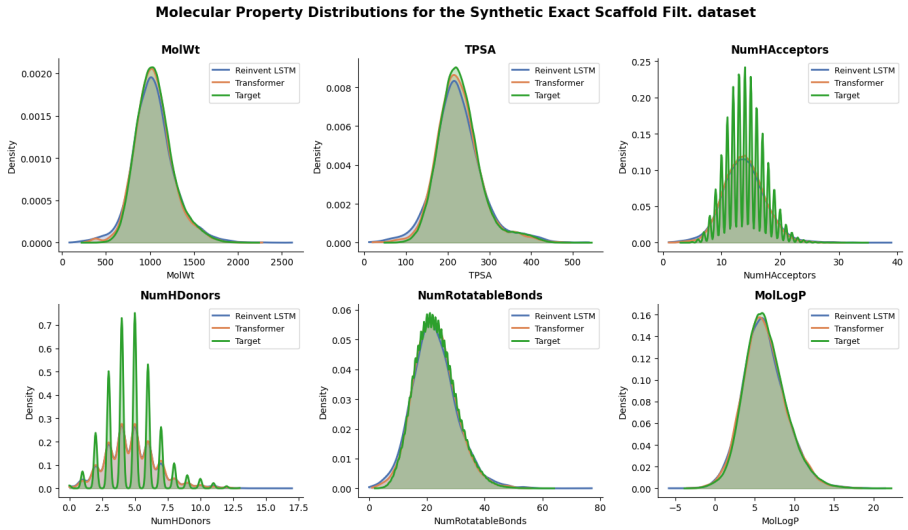


Figure B.19: The figure shows the distribution of generated PROTACs and their physicochemical properties. The figure visualizes the synthetic step of transfer learning

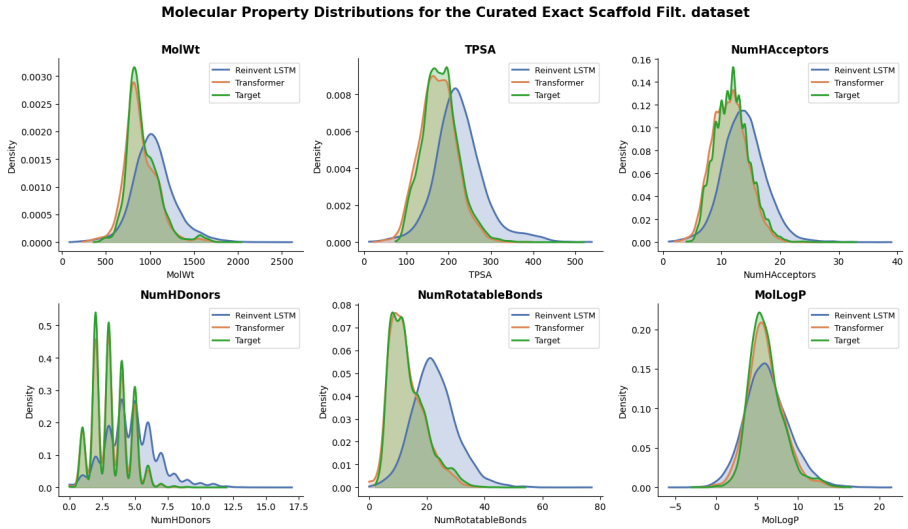


Figure B.20: The figure shows the distribution of generated PROTACs and their physicochemical properties. The figure visualizes the curated step of transfer learning

B.5.2 Similarity-Threshold dataset

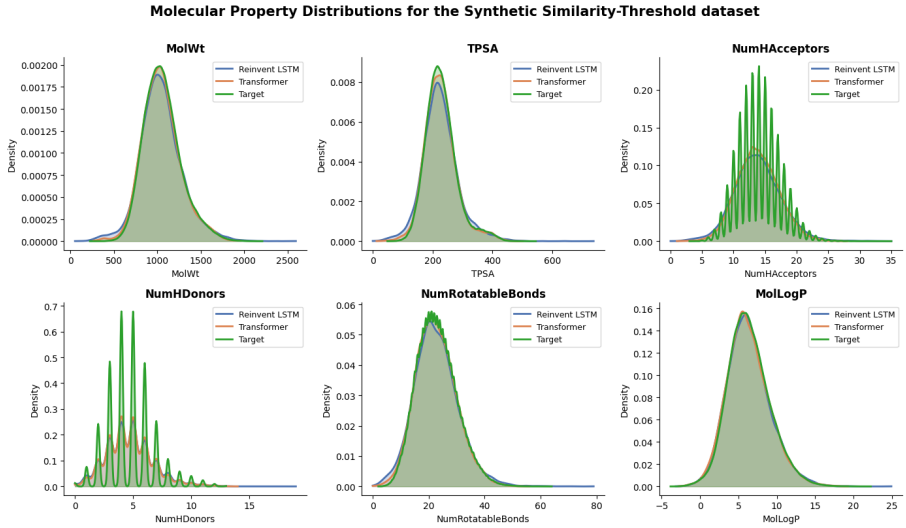


Figure B.21: The figure shows the distribution of generated PROTACs and their physicochemical properties. The figure visualizes the synthetic step of transfer learning

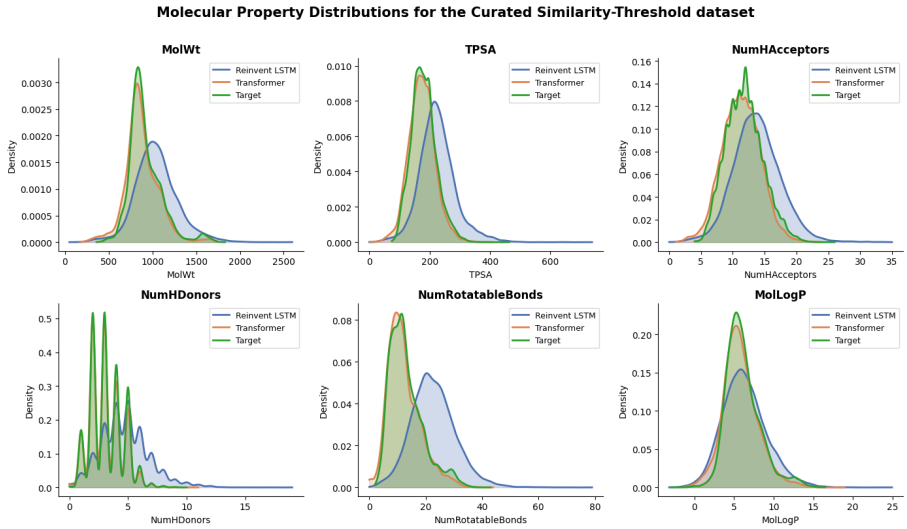


Figure B.22: The figure shows the distribution of generated PROTACs and their physicochemical properties. The figure visualizes the curated step of transfer learning

B.5.3 No AR

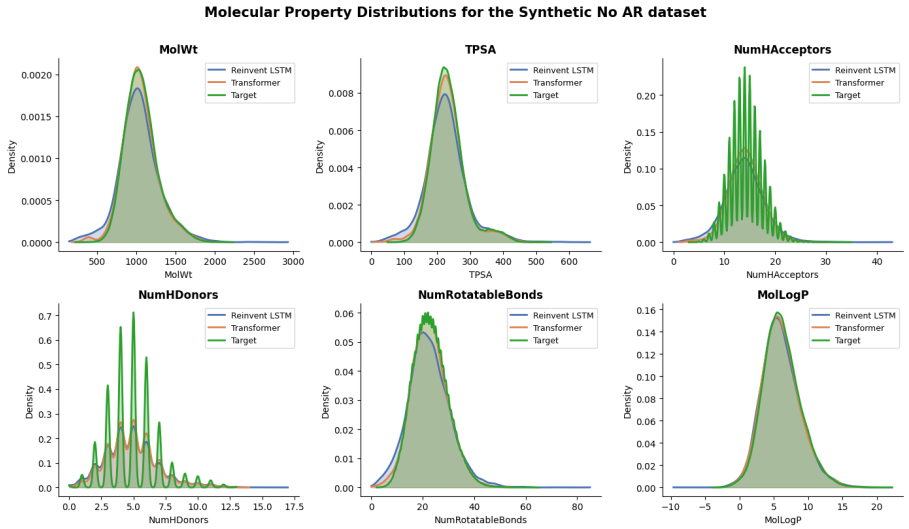


Figure B.23: The figure shows the distribution of generated PROTACs and their physicochemical properties. The figure visualizes the synthetic step of transfer learning

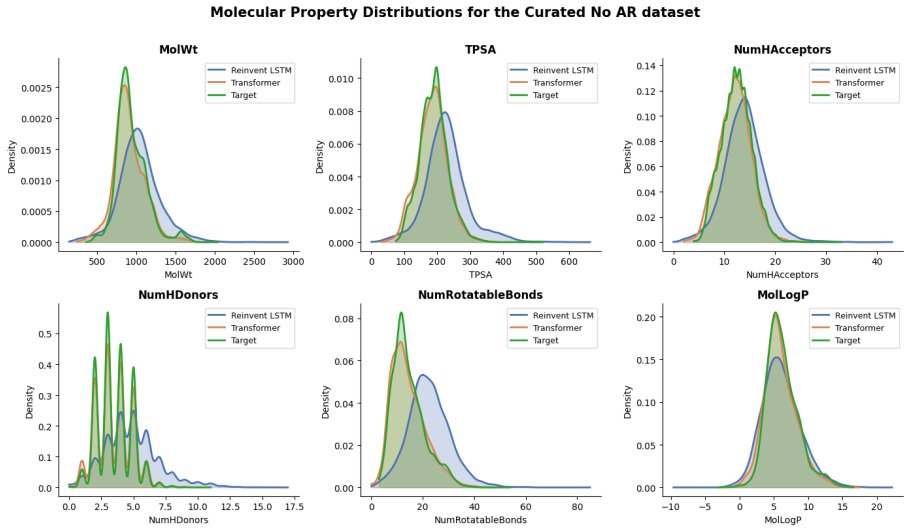


Figure B.24: The figure shows the distribution of generated PROTACs and their physicochemical properties. The figure visualizes the curated step of transfer learning

C

Appendix 3

C.1 Statistical Variance Analysis of Pipeline Robustness

This section provides a statistical evaluation of the results and the difference in the results between configurations. ANOVA is a frequently used statistical method for determining if there exists a statistically significant difference between a group of 3 or more classes [73]. Given a number of samples for each class, ANOVA defines a hypothesis test using the null hypothesis and alternative hypothesis presented in equation C.1. The null hypothesis states that for a given number of classes, there is no difference in their means, whereas the alternative hypothesis states that there is at least one class that does not share the mean with all other classes. ANOVA assumes that the data is drawn from a normal distribution, the variance between groups is homoscedastic to one order of magnitude, and that the classes are independent from each other [74]. Otherwise, the significance might be overstated.

$$\begin{aligned} H_0 &: \mu_0 = \mu_1 = \cdots = \mu_n \\ H_A &: \exists\, i, j \text{ such that } \mu_i \neq \mu_j \end{aligned} \tag{C.1}$$

In order to calculate if there is significant evidence to reject the null hypothesis, ANOVA uses the F-test presented in equation C.2. Given a number, n_i, samples for each class, ANOVA calculates the F-score by comparing the between-class variance with the intra-class variance. The between-class variance is calculated based on the class' internal mean compared to the grand mean, $\hat{Y}$ of all classes combined. K describes the degrees of freedom in the system, and N is the total number of samples across all classes.

$$F = \frac{\text{between class variance}}{\text{within class variance}} = \frac{\sum_{i=1}^{K} n_i(\bar{Y_i} - \hat{Y})^2/(K-1)}{\sum_{i,j}^{n}(Y_{i,j} - \bar{Y_i})^2/(N-K)} \tag{C.2}$$

Each table presents the grand mean, standard deviation, η^2 describing how much of the variance is observed in the system, F-score, and the p-value. The stability is defined accordingly:

- $\eta^2 \in [0, 0.01)$: Highly stable
- $\eta^2 \in [0.01, 0.06)$: Stable
- $\eta^2 \in [0.06, 0.14)$: Moderate Variance
- $\eta^2 \in [0.14, 1]$: High Variance

C.1.1 Statistical evaluation of pipeline robustness

C.1.1.1 Reinvent

Step	Metric	Grand Mean	Std Dev	η^2	F-score	p-value	Stability
TL synthetic	Activity	0.4912	0.1495	0.0000	0.5438	0.7036	Highly Stable
	DC50 (nM)	86.8887	125.3450	0.0000	0.3733	0.8279	Highly Stable
	Dmax (%)	78.0006	9.0764	0.0001	0.7157	0.5810	Highly Stable
	MolWt	1040.1853	240.0973	0.0001	0.6149	0.6519	Highly Stable
	TPSA	225.6454	59.4028	0.0000	0.1073	0.9800	Highly Stable
	NumHAcceptors	13.8378	3.6710	0.0000	0.2173	0.9289	Highly Stable
	NumHDonors	4.7283	1.9999	0.0000	0.3842	0.8201	Highly Stable
	NumRotatableBonds	22.5851	7.7197	0.0000	0.5204	0.7208	Highly Stable
	MolLogP	6.2217	2.7213	0.0001	1.6862	0.1500	Highly Stable
TL curated	Activity	0.5681	0.1915	0.0000	0.2356	0.9184	Highly Stable
	DC50 (nM)	50.2281	74.7409	0.0002	1.8191	0.1220	Highly Stable
	Dmax (%)	81.0067	8.6144	0.0000	0.5419	0.7050	Highly Stable
	MolWt	875.6552	226.8031	0.0000	0.3702	0.8301	Highly Stable
	TPSA	174.2772	52.6774	0.0001	0.5595	0.6921	Highly Stable
	NumHAcceptors	11.3185	3.4322	0.0001	0.7398	0.5647	Highly Stable
	NumHDonors	3.0783	1.4853	0.0001	0.8739	0.4785	Highly Stable
	NumRotatableBonds	13.0445	6.7016	0.0001	0.6794	0.6061	Highly Stable
	MolLogP	5.7874	2.3431	0.0000	0.2246	0.9248	Highly Stable
RL	Activity	0.7431	0.1581	0.0284	355.8473	<0.0001	Stable
	DC50 (nM)	34.9677	46.5070	0.0067	82.3313	<0.0001	Highly Stable
	Dmax (%)	85.4245	6.0708	0.0233	289.6037	<0.0001	Stable
	MolWt	878.0748	169.2164	0.0219	272.0636	<0.0001	Stable
	TPSA	168.8644	42.5618	0.0160	197.0497	<0.0001	Stable
	NumHAcceptors	11.0792	2.7129	0.0663	862.8047	<0.0001	Moderate Variance
	NumHDonors	2.8868	1.2409	0.0656	852.7702	<0.0001	Moderate Variance
	NumRotatableBonds	13.8763	5.1550	0.0563	724.6848	<0.0001	Stable
	MolLogP	5.9024	2.0070	0.0207	257.1083	<0.0001	Stable

Table C.1: ANOVA analysis of the REINVENT LSTM stability across identical runs for the "Exact Scaffold Filt." dataset.

Step	Metric	Grand Mean	Std Dev	η^2	F-score	p-value	Stability
TL synthetic	Activity	0.4967	0.1481	0.0000	0.3814	0.8221	Highly Stable
	DC50 (nM)	88.1936	127.8715	0.0001	1.0489	0.3802	Highly Stable
	Dmax (%)	78.0384	9.1209	0.0000	0.3255	0.8610	Highly Stable
	MolWt	1043.5161	244.4470	0.0001	1.6077	0.1692	Highly Stable
	TPSA	225.9267	60.0997	0.0003	3.2100	0.0121	Highly Stable
	NumHAcceptors	13.7763	3.6556	0.0001	1.2229	0.2986	Highly Stable
	NumHDonors	4.7353	2.0466	0.0002	2.0950	0.0786	Highly Stable
	NumRotatableBonds	22.6520	7.7437	0.0001	1.3575	0.2460	Highly Stable
	MolLogP	6.2937	2.7847	0.0001	1.7275	0.1407	Highly Stable
TL curated	Activity	0.5749	0.1882	0.0001	0.8673	0.4826	Highly Stable
	DC50 (nM)	48.0933	74.7776	0.0001	1.1065	0.3514	Highly Stable
	Dmax (%)	80.9646	8.6346	0.0000	0.2432	0.9139	Highly Stable
	MolWt	867.2996	231.1876	0.0000	0.2702	0.8973	Highly Stable
	TPSA	172.3867	52.7398	0.0000	0.2320	0.9205	Highly Stable
	NumHAcceptors	11.1366	3.4282	0.0000	0.2500	0.9098	Highly Stable
	NumHDonors	3.0403	1.4687	0.0000	0.2794	0.8915	Highly Stable
	NumRotatableBonds	12.5606	6.3449	0.0001	0.5645	0.6884	Highly Stable
	MolLogP	5.7662	2.3940	0.0001	0.6119	0.6541	Highly Stable
RL	Activity	0.7664	0.1359	0.1241	1733.7515	<0.0001	Moderate Variance
	DC50 (nM)	33.3507	35.9744	0.0351	444.4228	<0.0001	Stable
	Dmax (%)	86.0382	5.8131	0.2093	3238.7418	<0.0001	High Variance
	MolWt	887.5863	140.8692	0.0580	752.7710	<0.0001	Stable
	TPSA	164.8429	40.5298	0.1743	2582.4804	<0.0001	High Variance
	NumHAcceptors	10.2947	2.8852	0.4258	9071.6683	<0.0001	High Variance
	NumHDonors	3.2241	1.2766	0.1106	1520.7231	<0.0001	Moderate Variance
	NumRotatableBonds	15.0206	5.2887	0.2392	3844.8680	<0.0001	High Variance
	MolLogP	6.1980	2.1111	0.1456	2084.2022	<0.0001	High Variance

Table C.2: ANOVA analysis of the REINVENT LSTM stability across identical runs for the "Sim-Threshold" dataset.

Step	Metric	Grand Mean	Std Dev	η^2	F-score	p-value	Stability
TL synthetic	Activity	0.5047	0.1460	0.0001	1.3393	0.2526	Highly Stable
	DC50 (nM)	89.6660	122.4317	0.0000	0.5272	0.7157	Highly Stable
	Dmax (%)	78.3315	8.5205	0.0002	1.8226	0.1213	Highly Stable
	MolWt	1041.7555	264.5482	0.0000	0.3468	0.8464	Highly Stable
	TPSA	228.3619	65.2789	0.0001	1.1412	0.3349	Highly Stable
	NumHAcceptors	13.9907	3.9324	0.0000	0.4529	0.7704	Highly Stable
	NumHDonors	4.8635	2.1122	0.0001	1.0610	0.3740	Highly Stable
	NumRotatableBonds	22.4998	8.0383	0.0001	0.6106	0.6550	Highly Stable
	MolLogP	6.0488	2.8229	0.0000	0.5311	0.7129	Highly Stable
TL curated	Activity	0.5786	0.1812	0.0001	0.7601	0.5511	Highly Stable
	DC50 (nM)	55.9335	82.9163	0.0001	0.8606	0.4867	Highly Stable
	Dmax (%)	81.1701	8.2157	0.0000	0.4893	0.7437	Highly Stable
	MolWt	890.4938	244.0115	0.0002	1.9506	0.0991	Highly Stable
	TPSA	181.0766	55.8654	0.0003	3.1326	0.0138	Highly Stable
	NumHAcceptors	11.7481	3.6453	0.0003	2.7378	0.0271	Highly Stable
	NumHDonors	3.3835	1.4691	0.0002	2.1592	0.0709	Highly Stable
	NumRotatableBonds	13.9506	6.8648	0.0002	2.2119	0.0650	Highly Stable
	MolLogP	5.8302	2.5315	0.0001	1.0868	0.3611	Highly Stable
RL	Activity	0.7520	0.1513	0.0614	793.2064	<0.0001	Moderate Variance
	DC50 (nM)	34.6988	47.1656	0.0159	195.7474	<0.0001	Stable
	Dmax (%)	86.1953	6.1571	0.0649	841.3036	<0.0001	Moderate Variance
	MolWt	901.2156	185.4374	0.0426	539.4719	<0.0001	Stable
	TPSA	172.2749	47.4308	0.0437	554.2864	<0.0001	Stable
	NumHAcceptors	11.2614	3.0047	0.0751	985.2532	<0.0001	Moderate Variance
	NumHDonors	3.1060	1.2217	0.0399	504.4424	<0.0001	Stable
	NumRotatableBonds	13.4603	5.4391	0.0697	908.2027	<0.0001	Moderate Variance
	MolLogP	6.1746	2.2008	0.0634	820.7850	<0.0001	Moderate Variance

Table C.3: ANOVA analysis of REINVENT LSTM stability across identical runs for the "No AR" dataset.

C.1.1.2 Transformer

Step	Metric	Grand Mean	Std Dev	η^2	F-score	p-value	Stability
TL synthetic	Activity	0.5004	0.1478	0.0002	2.7357	0.0272	Highly Stable
	DC50 (nM)	78.8073	102.6708	0.0002	1.9569	0.0981	Highly Stable
	Dmax (%)	78.3016	9.0813	0.0000	0.0864	0.9867	Highly Stable
	MolWt	1048.3236	211.5748	0.0003	3.6523	0.0056	Highly Stable
	TPSA	227.4315	51.4458	0.0002	2.2962	0.0567	Highly Stable
	NumHAcceptors	13.9820	3.2177	0.0005	5.7453	0.0001	Highly Stable
	NumHDonors	4.7378	1.9013	0.0000	0.0459	0.9960	Highly Stable
	NumRotatableBonds	22.8349	7.0599	0.0001	0.5824	0.6754	Highly Stable
	MolLogP	6.2653	2.6371	0.0000	0.3989	0.8095	Highly Stable
TL curated	Activity	0.6089	0.1806	0.0002	1.7273	0.1408	Highly Stable
	DC50 (nM)	42.2072	69.2050	0.0004	3.6743	0.0054	Highly Stable
	Dmax (%)	82.7754	8.1205	0.0018	16.1620	<0.0001	Highly Stable
	MolWt	886.6298	182.4081	0.0001	0.7557	0.5541	Highly Stable
	TPSA	175.2456	42.3391	0.0010	9.0631	<0.0001	Highly Stable
	NumHAcceptors	11.4244	2.9204	0.0011	10.1315	<0.0001	Highly Stable
	NumHDonors	3.1211	1.3448	0.0002	1.3999	0.2311	Highly Stable
	NumRotatableBonds	13.1181	5.9467	0.0000	0.0724	0.9905	Highly Stable
	MolLogP	5.9859	2.1629	0.0011	10.0633	<0.0001	Highly Stable
RL	Activity	0.7349	0.1427	0.2381	1940.7563	<0.0001	High Variance
	DC50 (nM)	23.7314	21.0126	0.1498	1093.7595	<0.0001	High Variance
	Dmax (%)	91.7812	4.5311	0.2147	1697.4986	<0.0001	High Variance
	MolWt	940.4546	177.2253	0.3932	4023.1446	<0.0001	High Variance
	TPSA	150.9577	41.1689	0.3622	3527.0558	<0.0001	High Variance
	NumHAcceptors	10.8060	2.5081	0.3758	3738.4141	<0.0001	High Variance
	NumHDonors	3.6355	1.8313	0.4698	5503.3024	<0.0001	High Variance
	NumRotatableBonds	19.2381	6.0512	0.2839	2462.3180	<0.0001	High Variance
	MolLogP	8.7916	2.1256	0.2069	1619.5832	<0.0001	High Variance

Table C.4: ANOVA analysis of the Transformer stability across identical runs for the "Exact Scaffold Filt." dataset.

Step	Metric	Grand Mean	Std Dev	η^2	F-score	p-value	Stability
TL synthetic	Activity	0.5063	0.1469	0.0003	3.1422	0.0136	Highly Stable
	DC50 (nM)	82.1227	112.4114	0.0000	0.0034	1.0000	Highly Stable
	Dmax (%)	78.4590	9.0477	0.0002	1.7171	0.1430	Highly Stable
	MolWt	1050.3156	218.5938	0.0001	1.6738	0.1529	Highly Stable
	TPSA	227.2146	52.9564	0.0001	0.7081	0.5862	Highly Stable
	NumHAcceptors	13.9228	3.2511	0.0001	1.3898	0.2346	Highly Stable
	NumHDonors	4.7098	1.9515	0.0000	0.0605	0.9932	Highly Stable
	NumRotatableBonds	22.8590	7.0674	0.0001	1.3092	0.2639	Highly Stable
	MolLogP	6.3351	2.7259	0.0002	2.2416	0.0620	Highly Stable
TL curated	Activity	0.6049	0.1805	0.0000	0.0012	1.0000	Highly Stable
	DC50 (nM)	41.1624	65.6009	0.0001	0.5603	0.6915	Highly Stable
	Dmax (%)	82.5866	8.0154	0.0003	2.3801	0.0493	Highly Stable
	MolWt	874.9503	194.9626	0.0000	0.0449	0.9962	Highly Stable
	TPSA	172.7526	44.2181	0.0001	0.9113	0.4561	Highly Stable
	NumHAcceptors	11.2357	3.0121	0.0000	0.3383	0.8523	Highly Stable
	NumHDonors	3.1078	1.3674	0.0002	1.9178	0.1044	Highly Stable
	NumRotatableBonds	12.5663	5.7815	0.0001	0.8290	0.5064	Highly Stable
	MolLogP	5.9143	2.2130	0.0004	3.7131	0.0050	Highly Stable
RL	Activity	0.7276	0.1750	0.0770	596.3829	<0.0001	Moderate Variance
	DC50 (nM)	27.8702	26.0637	0.0268	197.2675	<0.0001	Stable
	Dmax (%)	89.7169	5.5390	0.1065	852.5014	<0.0001	Moderate Variance
	MolWt	908.5101	171.3227	0.2975	3027.9024	<0.0001	High Variance
	TPSA	151.9138	42.0558	0.3779	4343.4208	<0.0001	High Variance
	NumHAcceptors	10.6580	2.4899	0.3318	3549.9752	<0.0001	High Variance
	NumHDonors	3.4548	1.4634	0.3981	4729.5804	<0.0001	High Variance
	NumRotatableBonds	16.8360	5.6393	0.4440	5711.2567	<0.0001	High Variance
	MolLogP	7.3203	1.9126	0.1576	1337.4504	<0.0001	High Variance

Table C.5: ANOVA analysis of the Transformer LSTM stability across identical runs for the "Sim-Threshold" dataset.

Step	Metric	Grand Mean	Std Dev	η^2	F-score	p-value	Stability
TL synthetic	Activity	0.5212	0.1391	0.0002	2.6276	0.0327	Highly Stable
	DC50 (nM)	79.3274	98.3926	0.0001	0.9820	0.4158	Highly Stable
	Dmax (%)	78.9661	8.2812	0.0002	2.6343	0.0323	Highly Stable
	MolWt	1061.2975	217.9935	0.0006	6.2339	0.0001	Highly Stable
	TPSA	232.0404	53.1241	0.0002	2.1726	0.0693	Highly Stable
	NumHAcceptors	14.3102	3.1881	0.0005	5.5087	0.0002	Highly Stable
	NumHDonors	4.8700	1.9173	0.0000	0.0617	0.9930	Highly Stable
	NumRotatableBonds	22.8570	6.7906	0.0001	1.1095	0.3500	Highly Stable
	MolLogP	6.1952	2.7405	0.0004	4.5110	0.0012	Highly Stable
TL curated	Activity	0.6002	0.1723	0.0005	4.0240	0.0029	Highly Stable
	DC50 (nM)	49.1775	93.4072	0.0002	1.4457	0.2160	Highly Stable
	Dmax (%)	82.3843	7.7643	0.0001	0.5271	0.7159	Highly Stable
	MolWt	905.4030	200.6499	0.0001	0.6932	0.5965	Highly Stable
	TPSA	184.6636	45.3899	0.0003	2.6355	0.0322	Highly Stable
	NumHAcceptors	12.0612	3.1534	0.0005	4.2212	0.0020	Highly Stable
	NumHDonors	3.4878	1.3233	0.0004	3.7061	0.0051	Highly Stable
	NumRotatableBonds	14.4155	6.5806	0.0001	1.1619	0.3254	Highly Stable
	MolLogP	5.9702	2.3621	0.0000	0.0016	1.0000	Highly Stable
RL	Activity	0.8002	0.1080	0.1165	501.0515	<0.0001	Moderate Variance
	DC50 (nM)	27.5668	15.6834	0.0152	58.6240	<0.0001	Stable
	Dmax (%)	91.2665	4.4371	0.2756	1446.3934	<0.0001	High Variance
	MolWt	821.5588	94.5995	0.3536	2079.9394	<0.0001	High Variance
	TPSA	115.5039	35.0239	0.5759	5162.7811	<0.0001	High Variance
	NumHAcceptors	7.9287	2.1980	0.6067	5863.6807	<0.0001	High Variance
	NumHDonors	3.4567	1.0060	0.1035	438.8127	<0.0001	Moderate Variance
	NumRotatableBonds	12.7986	3.2934	0.2028	966.8096	<0.0001	High Variance
	MolLogP	7.8839	1.4763	0.1209	522.9801	<0.0001	Moderate Variance

Table C.6: ANOVA analysis of REINVENT LSTM stability across identical runs for the "No AR" dataset.

C.1.2 Statistical evaluation of best runs

C.1.2.1 Reinvent

Step	Metric	Grand Mean	Std Dev	η^2	F-score	p-value	Stability
Synthetic	Activity	0.4994	0.1484	0.0016	23.5596	<0.0001	Highly Stable
	DC50 (nM)	88.3393	128.2736	0.0001	0.9299	0.3946	Highly Stable
	Dmax (%)	78.2314	8.8763	0.0006	8.4868	0.0002	Highly Stable
	MolWt	1042.4464	252.0725	0.0000	0.5416	0.5818	Highly Stable
	TPSA	226.3357	62.0411	0.0002	2.2491	0.1055	Highly Stable
	NumHAcceptors	13.8762	3.7950	0.0003	4.8081	0.0082	Highly Stable
	NumHDonors	4.7635	2.0542	0.0005	7.5269	0.0005	Highly Stable
	NumRotatableBonds	22.6196	7.8944	0.0002	2.4252	0.0885	Highly Stable
	MolLogP	6.2260	2.7825	0.0014	20.6468	<0.0001	Highly Stable
Curated	Activity	0.5752	0.1865	0.0006	7.6397	0.0005	Highly Stable
	DC50 (nM)	50.8323	73.2601	0.0025	32.6473	<0.0001	Highly Stable
	Dmax (%)	81.0593	8.4840	0.0002	2.3531	0.0951	Highly Stable
	MolWt	878.3090	234.1168	0.0018	23.8065	<0.0001	Highly Stable
	TPSA	176.0231	54.2342	0.0050	65.5795	<0.0001	Highly Stable
	NumHAcceptors	11.4140	3.5329	0.0053	69.8441	<0.0001	Highly Stable
	NumHDonors	3.1734	1.4907	0.0108	143.5902	<0.0001	Stable
	NumRotatableBonds	13.2185	6.6700	0.0081	106.9715	<0.0001	Highly Stable
	MolLogP	5.8083	2.4141	0.0003	4.3527	0.0129	Highly Stable
RL	Activity	0.7675	0.1470	0.0005	7.9813	0.0003	Highly Stable
	DC50 (nM)	29.8286	39.1649	0.0022	32.2570	<0.0001	Highly Stable
	Dmax (%)	86.5168	5.7043	0.0091	133.3865	<0.0001	Highly Stable
	MolWt	871.4314	152.4910	0.0366	553.8589	<0.0001	Stable
	TPSA	166.8021	42.2926	0.0356	538.8619	<0.0001	Stable
	NumHAcceptors	10.6474	2.9286	0.1850	3311.8460	<0.0001	High Variance
	NumHDonors	3.2881	1.3553	0.0205	305.9100	<0.0001	Stable
	NumRotatableBonds	12.3575	4.3788	0.0386	586.2252	<0.0001	Stable
	MolLogP	5.6758	1.8474	0.0317	477.9378	<0.0001	Stable

Table C.7: ANOVA analysis between the different datasets for the training steps. The analysis is based on the performance across metrics for the REINVENT LSTM.

C.1.2.2 Transformer

Step	Metric	Grand Mean	Std Dev	η^2	F-score	p-value	Stability
Synthetic	Activity	0.5045	0.1457	0.0038	47.8810	<0.0001	Highly Stable
	DC50 (nM)	81.7165	106.7501	0.0000	0.1901	0.8268	Highly Stable
	Dmax (%)	78.3913	8.8547	0.0006	7.6447	0.0005	Highly Stable
	MolWt	1045.2490	219.3017	0.0004	4.5568	0.0105	Highly Stable
	TPSA	227.5667	53.5725	0.0015	18.8153	<0.0001	Highly Stable
	NumHAcceptors	13.9460	3.2850	0.0021	27.2491	<0.0001	Highly Stable
	NumHDonors	4.7744	1.9531	0.0015	19.1518	<0.0001	Highly Stable
	NumRotatableBonds	22.7157	7.0537	0.0000	0.0296	0.9708	Highly Stable
	MolLogP	6.1905	2.7169	0.0008	10.3732	<0.0001	Highly Stable
Curated	Activity	0.6046	0.1780	0.0008	7.6201	0.0005	Highly Stable
	DC50 (nM)	43.3109	58.1516	0.0040	40.5036	<0.0001	Highly Stable
	Dmax (%)	82.5694	7.9940	0.0013	12.7275	<0.0001	Highly Stable
	MolWt	888.5687	193.0292	0.0047	47.7878	<0.0001	Highly Stable
	TPSA	177.6759	43.9293	0.0137	140.8554	<0.0001	Stable
	NumHAcceptors	11.5756	3.0381	0.0144	147.7445	<0.0001	Stable
	NumHDonors	3.2298	1.3498	0.0200	206.0615	<0.0001	Stable
	NumRotatableBonds	13.3289	6.1315	0.0182	187.4142	<0.0001	Stable
	MolLogP	5.9393	2.2466	0.0015	14.8063	<0.0001	Highly Stable
RL	Activity	0.7780	0.1366	0.0695	338.2299	<0.0001	Moderate Variance
	DC50 (nM)	20.9663	18.2263	0.0272	126.4808	<0.0001	Stable
	Dmax (%)	91.8458	4.3726	0.1759	967.2037	<0.0001	High Variance
	MolWt	804.5939	128.9160	0.1143	584.5447	<0.0001	Moderate Variance
	TPSA	116.6089	31.6764	0.0673	326.9531	<0.0001	Moderate Variance
	NumHAcceptors	8.8629	1.6818	0.0138	63.3153	<0.0001	Stable
	NumHDonors	1.9436	1.3271	0.0244	113.5499	<0.0001	Stable
	NumRotatableBonds	15.0279	4.2648	0.1349	706.6749	<0.0001	Moderate Variance
	MolLogP	7.8414	1.8936	0.0920	458.8613	<0.0001	Moderate Variance

Table C.8: ANOVA analysis between the different datasets for the training steps. The analysis is based on the performance across metrics for the REINVENT LSTM.

D

Appendix 4

D.1 Reproducibility

The following tables show the hyperparameters used for training both models for all stages and all datasets. The hyperparameters presented below were found using Optuna.

Dataset	Training step	Learning rate	Gamma	Step	Min learning rate	Epochs
Exact Scaffold Filt.	TL: Synthetic	$1.0158298384121807e-06$	0.946729387814737	2	0.0	40
	TL: Curated	$2.4047549058215508e-05$	0.9288422078116606	2	0.0	100
Similarity-Threshold	TL: Synthetic	$1.7471742617103054e-06$	0.7961914442108753	4	0.0	40
	TL: Curated	$5.946329971283536e-05$	0.9005083110090645	1	0.0	100
No AR	TL: Synthetic	$6.811100479240774e-07$	0.9616719667385938	5	0.0	40
	TL: Curated	$9.00933337639123e-05$	0.7621422563898215	2	0.0	100

Table D.1: REINVENT LSTM hyperparameters for transfer learning.

Dataset	Training step	Rate	Sigma	Min steps	Max steps	Loss function
Exact Scaffold Filt.	RL	0.00011605135527443305	57	200	300	dap
Similarity-Threshold	RL	0.00016300864383799458	58	200	300	dap
No AR	RL	$7.861669606308871e-05$	59	200	300	dap

Table D.2: REINVENT LSTM hyperparameters for reinforcement learning.

Dataset	Training step	Learning rate	Warmup ratio	Weight decay	LR Scheduler	Beta (GRPO only)	Epochs
Full	TL: Synthetic	$4.654332843228168e-06$	0.07435438220182458	0.005817761345828483	Cosine	–	40
	TL: Curated	0.0005543400177676147	0.011982341439028377	0.006720129599755071	Cosine	–	130
	RL	$6.422838547724939e-05$	0.011345671706686721	0.06301528733819535	Cosine	0.06079916463759706	8
Similarity-threshold	TL: Synthetic	$7.5330281525420445e-06$	0.054361467011771684	0.08925787363651803	Cosine	–	40
	TL: Curated	0.00021475504707047747	0.008286722435867538	0.00013229135437582045	Linear	–	130
	RL	$1.0595199144311133e-05$	0.06809129100117817	0.0006587696495877953	Cosine	0.024970580029881978	8
no AR target POI	TL: Synthetic	$5.23973166289907e-06$	0.019342934736704273	0.04878064564919768	Cosine	–	40
	TL: Curated	0.0006483840228283588	0.014219757524391474	0.010698657132480405	Cosine	–	130
	RL	$1.4615352925401675e-05$	0.002781345588135242	0.09887914468923807	Cosine	0.02932883422620682	8

Table D.3: Transformer hyperparameters.

D.2 Availablity

The pre-trained LSTM model used for this thesis was provided by Hannes Löffler at AstraZeneca and is available at:
`https://drive.google.com/file/d/1rOwWruKtWHo5jyJ7s1-4guOSg_IiX_OK/view?usp=sharing`
The pre-trained Transformer used for this thesis was provided by Chitsaz *et al.* [10] and is available at:
`https://huggingface.co/chandar-lab/NovoMolGen_32M_SMILES_AtomWise`
The datasets used to train the models in this thesis are available at:

- Synthetic: `https://zenodo.org/records/15797310`
- Curated: `https://huggingface.co/datasets/ailab-bio/TACK`
- TPDdb: `https://tpddb.idrblab.net/`

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden
www.chalmers.se

CHALMERS
UNIVERSITY OF TECHNOLOGY