Augmenting Generative De Novo Drug Design with Synthetic Data

By:

Hanz Tantiangco

A thesis submitted to the University of Sheffield in fulfilment of the requirements to the degree of Doctor of Philosophy

This work was sponsored by:

The University of Sheffield

Information School – Faculty of Social Sciences

April 2026

Acknowledgements

Firstly, I would like to thank my supervisors Prof. Val Gillet, Prof. Beining Chen, and Dr James Webster: Val for her invaluable mentorship, support, and patience; Beining for her advice and insights into medicinal chemistry; James for his constant guidance and his fountain of ideas. The tireless encouragement and push from my supervisors throughout my PhD helped me reach a goal I thought would be too challenging for me.

I would also like to thank my colleagues from the Chemoinformatics Research Group who also have supported me through this journey and provided lively discussion of chemoinformatics: Jess S., Moritz, James, Zied, Savins, Terence, Emma, Toby, and Hannah. Additionally, the colleagues from the wider Information School PGR Group, especially Zenan and Jess F. Furthermore, I would like to thank my family and friends who have supported me during this journey.

Finally, I would like to thank Dr. Antonio de la Vega de León for initiating the PhD project and the Engineering and Physical Sciences Research Council (EPSRC) for their funding and support.

Abstract

A recent trend in de novo drug design is the use of generative deep learning models, such as recurrent neural networks (RNNs). However, despite the promises of RNNs, they can be outperformed by traditional methods such as the Graph GA (Brown *et al.*, 2019). Recent works have used randomised/enumerated SMILES as a form of data augmentation to expand the training data. This has been shown to improve the performance of RNNs for distribution learning tasks (Arús-Pous, Johansson, *et al.*, 2019; Moret *et al.*, 2020). However, the application of data augmentation in goal-directed generative design remains limited. This thesis describes the development of novel operators for the data augmentation of generative *de novo* design for goal-directed generation. The first step was to identify whether data augmentation of the primary/pretraining dataset improved model performance. Results indicate that, in general, introducing data augmentation to the primary training dataset improved goal-directed generation performance.

Next, data augmentation was investigated at different points of the fine-tuning loop in an RNN-LSTM. It was identified that subset loop augmentation performed the best. Subsequently, NLP-inspired operators were developed, which include atom-swap, atom-insert, atom-delete, and atom-fusion operators. Although these operators resulted in improvements of the design objectives, it was identified that they had a negative impact on the synthesisability of the generated molecules. Finally, methods to improve the synthesisability of the generated molecules were explored. These methods include chemistry-based operators (graph-fusion and bioisosteric substitution), masked atom-fusion (SAscore based- and functional group-based masking), and finetuning dataset filtering. Results show that bioisosteric substitution may perform the best overall as it is able to significantly increase the MPO score with minimal impact on the SAscore of the generated molecules relative to the other operators. Additionally, filtering of the finetuning dataset generally improved the SAscore of the generated molecules.

Table of Contents

List of Figures

List of Tables

Table of Common Abbreviations

AEs - Autoencoders

AIS - Atom-in-SMILES

AP - Atom-Pair

ASCII - American Standard Code for Information Interchange

BM - Bemis-Murcko

BRICS – Breaking of Retrosynthetically Interesting Chemical Substructures

CADD - Computer-aided Drug Design

CLMs – Chemical Language Models

CV – Computer Vision

DL - Deep Learning

DMTA – Design-Make-Test-Analyse

DNNs - Deep Neural Networks

ECFC - Extended Connectivity Fragment Count)

eSMILES - Encoded SMILES

GA - Genetic Algorithms

GANs - Generative Adversarial Network

GDBs - Generated Databases

Graph-GA – Graph Genetic Algorithm

LD - Lavenshtein Distance

logP - Octanol-water Partition Coefficient

ML - Machine Learning

MNIST - Modified National Institute of Standards and Technology

MPO - Multi-property Objective

NLP – Natural Language Processing

PAINS - Pan-assay Interference Compounds

QED - Quantitative Estimate of Drug-likeness

QSAR – Quantitative Structure-Activity Relationship

QSPR - Quantitative Structure-property Relationship

RL - Reinforcement Learning

RNN - Recurrent Neural Network

RNN-LSTM – Recurrent Neural Network with Long Short-term Memory

SAscore – Synthetic Accessibility Score

SMILES - Simplified Molecular Input Line Entry System

SPP - Similar Property Principle

Tc - Tanimoto Coefficient

TPSA - topological surface area

VAEs - Variational autoencoders

VS - Virtual Screening

WLN - Wiswesser Line Notation

Chapter 1. Introduction

De novo drug design aims to generate novel compounds from scratch based on a set of desired properties. The core algorithm is composed of assembly, the process of building molecules; scoring, the process of evaluating the generated molecules; and optimisation, an iterative process of refinement to improve the scores of the generated molecules. Each component can be performed either separately or collectively (Schneider and Clark, 2019). Traditional *de novo* drug design approaches rely on explicit chemical knowledge to define chemical transformations and assembly rules, thus traditional *de novo* design methods are often called rule-based approaches. Traditional methods include genetic algorithms. However, the reliance on predefined rules is a limitation as it restricts chemical space exploration (Schneider *et al.*, 2020).

More recent methods rely on implicit modelling using generative deep learning, termed here as generative de novo design. The growing interest in deep learning can be attributed to the growing computational power available, the increasing availability of data, and successes in domains such as computer vision and natural language processing (Chen et al., 2018; Liu et al., 2021).

Deep learning methods are data-driven meaning that they learn directly from the data without pre-defined rules. The minimised need for manual feature engineering in deep learning has been suggested to overcome the inefficient and incomplete exploration of the chemical space made by traditional methods but with the risk of sacrificing synthesizability (Chen et al., 2018; Jiménez-Luna et al., 2021; Schneider et al., 2020).

The main disadvantage of deep learning-based methods is that they rely on large datasets from which to learn and this potentially limits their application for generative de novo design as drug discovery projects tend to be low-data regime problems. Data augmentation has been used to address the limitation of deep learning methods in the fields of computer vision (CV) and natural language processing (NLP) (Shorten and Khoshgoftaar, 2019; Shorten, Khoshgoftaar and Furht, 2021). As a result, data augmentation has been increasingly investigated in the field of computer-aided drug design, such as *de novo* design (Van Tilborg *et al.*, 2024). However, the methods for data augmentation applied to *de novo* design remain limited, especially for goal-directed generation tasks. Therefore, the aim of this thesis is to develop novel data augmentation operators for generative *de novo* design for goal-directed generation tasks.

The thesis is organised as follows, Chapter 2 provides an overview of drug discovery and the Design-Make-Test-Analyse (DMTA) cycle. It then describes computer-aided drug design and how it has been applied to the Design stage of the DMTA cycle. Key concepts covered include the chemical space, chemical space navigation, molecular representations and descriptors.

Chapter 3 outlines the core components of *de novo* design which include assembly, scoring and optimisation. Traditional approaches to de novo design are described before discussion of the more recently developed techniques which make use of deep learning technologies and are termed generative *de novo* design methods. The general workflow of generative *de novo* design is described followed by different architectures that have been developed, which includes generative adversarial networks (GANs), autoencoders, and chemical language models (CLMs).

Chapter 4 introduces the concept of data augmentation and its application to deep learning models. Existing data augmentation methods from the NLP domain are discussed. Finally, how data augmentation has been applied in chemoinformatics is explored.

The rest of the thesis describes the experimental work undertaken. Chapter 4 provides a description of the data and methods that have been used throughout the thesis. This includes the GuacaMol's benchmark and baseline models (RNN-LSTM and Graph-GA) (Brown *et al.*, 2019), the use of randomised SMILES as a baseline against which the novel data augmentation methods are evaluated, and the statistical significance test applied later.

Chapter 6 investigates the effect of data augmentation on goal-directed performance when applied to the primary dataset used to pretrain an RNN-LSTM's. The data augmentation methods used include duplication, randomised SMILES, and BRICS-based enumeration.

Chapter 7 describes the development of novel data augmentation operators for the SMILES representation that are inspired by methods from the NLP domain, termed NLP-inspired operators. These NLP-inspired operators include atom-swap, atom-insert, atom-delete, and atom-fusion. Additionally, data augmentation was applied to the secondary dataset, i.e. the dataset used during goal-directed generation. It was hypothesized that this direct approach should provide more benefit to goal-directed generation performance compared to augmenting the primary dataset.

Chapter 8 examines the synthesisability of the molecules generated by the augmented RNN-LSTM models. It then describes the development of data augmentation methods that aim to improve the synthesisability of the generated molecules. These methods include chemistry-based operators (graph-fusion and bioisosteric substitution), masked atom-level operators (SAscore- and functional group-based masking), and filtering the finetuning dataset.

The final experimental chapter, Chapter 9, explores the MolScore benchmark (Thomas *et al.*, 2024). The benchmark provides a more realistic evaluation of generative *de novo* design models compared to GuacaMol's similarity-based metrics, by providing pretrained QSAR models. The QSAR models investigated include single and multi-target optimisation tasks.

Chapter 10 concludes the thesis and discusses the limitations of the developed methods and potential future work.

Chapter 2. Drug Discovery with Computer-Aided Drug Design

2.1 Introduction

The chemical space encompasses the entire universe of all possible molecules, both known and unknown (Lipinski and Hopkins, 2004). In drug discovery projects, the aim is to identify biologically active molecules in this vast chemical space. These small molecules occupy a subset of the chemical space known as the "drug-like" chemical space. The size of the chemical space has been estimated to be in the order of 10^{60} (Bohacek, McMartin and Guida, 1996). Historically, searching this space relied on inefficient trial-and-error (experimental screening) or serendipity approaches (Ban, 2006; Poduri, 2021). Traversing the vast drug-like chemical space in such a manner is an expensive and time-intensive endeavour. On average it takes 12 years for a potential drug to be clinically approved and the whole process can potentially cost $3 billion and has a high attrition rate of ~90% (Vincent Rajkumar, 2020).

The latter part of the 20th century saw the development of computer-aided drug design (CADD) techniques. CADD techniques include analysis tools for data mining and idea generators for molecular design (Schneider and Baringhaus, 2008). These tools allowed a more efficient and rational approach when navigating the chemical space, helping minimise failure and costs (Macalino *et al.*, 2015). CADD has therefore become important in modern drug discovery. This chapter describes the drug discovery stage and what is meant by the chemical space and how it can be navigated with CADD.

2.2 The Drug Discovery Stages

The process of bringing a therapeutic drug to market involves drug discovery and development. Drug discovery is the first stage which involves the pre-discovery and drug discovery stages to identify potential candidate drugs, discussed further below. Compounds that successfully pass this stage then go through a drug development phase. The first step in this phase is a pre-clinical study, where compounds undergo testing in *in vitro* and *in vivo* studies to evaluate their pharmacological properties, toxicity, and safety. Drugs that pass through basic safety tests then undergo clinical trials, where the compounds are tested in humans in phases to assess the efficacy and safety of the drug. Finally, the regulatory and approval stage involves submission of the drug for approval for market sale (Singh *et al.*, 2023).

2.2.1 Pre-discovery Stage

The pre-discovery stage involves disease mechanism research and identifying potential drug targets. This stage can be broken down into target identification and target validation. Target identification involves studying the molecular basis of a disease to identify potential drug targets which may be in the form of proteins, genes, and molecules involved in the disease pathway. Methods include biochemical affinity purification and biomedical data mining using bioinformatics approaches (Schenone *et al.*, 2013; Katsila *et al.*, 2016; Tabana *et al.*, 2023). Target validation involves investigating the suitability of potential drug targets. This is typically done through chemical validation and/or investigations on how target modulation affects disease progression in a model of interest (Wyatt *et al.*, 2011).

2.2.2 Drug Discovery Stage

The drug discovery stage is where compounds and other therapeutic modalities are designed and tested. This stage includes hit identification, hit-to-lead, and lead optimisation (Hughes *et al.*, 2011).

Hit identification involves screening large chemical libraries to identify potential drug candidates that bind to the target. Examples of common screening methods include high-throughput screening, fragment-based screening, and virtual screening. Compounds that show potential activity against a specific target of interest from the screen, usually in the micromolar range of activity, are defined as hits (Walters and Namchuk, 2003; Ashraf *et al.*, 2024).

The hit-to-lead or lead generation stage involves more intensive investigation on the activity and selectivity of the hits from the initial screening stage. The aim is to have lead compounds, that is, compounds that have better properties than the initial screening hits, which can serve as a starting point for further optimisation. The lead compounds tend to be in the nanomolar range for activity (Bleicher *et al.*, 2003; Keserű and Makara, 2006).

Finally, lead optimisation involves the further refinement of the lead compounds. At this stage, lead compounds undergo cycles of structural modification. The aim here is to mitigate the unwanted properties whilst maintaining the favourable properties of the lead compounds to develop them into preclinical candidates. Properties investigated can include pharmacological profiling, pharmacokinetic optimisation, and safety assessment (Jorgensen, 2009; Verma and Pathak, 2022).

2.3 Design-Make-Test-Analyse (DMTA) cycle

The hit identification, hit-to-lead, and lead optimisation stage is commonly portrayed as a Design-Make-Test-Analyse (DMTA) cycle (Figure 2-1). The DMTA cycle is an iterative process where results from previous cycles are used to inform the next iterations until potential drug candidates are generated (Plowright *et al.*, 2012; Ghiandoni *et al.*, 2024).

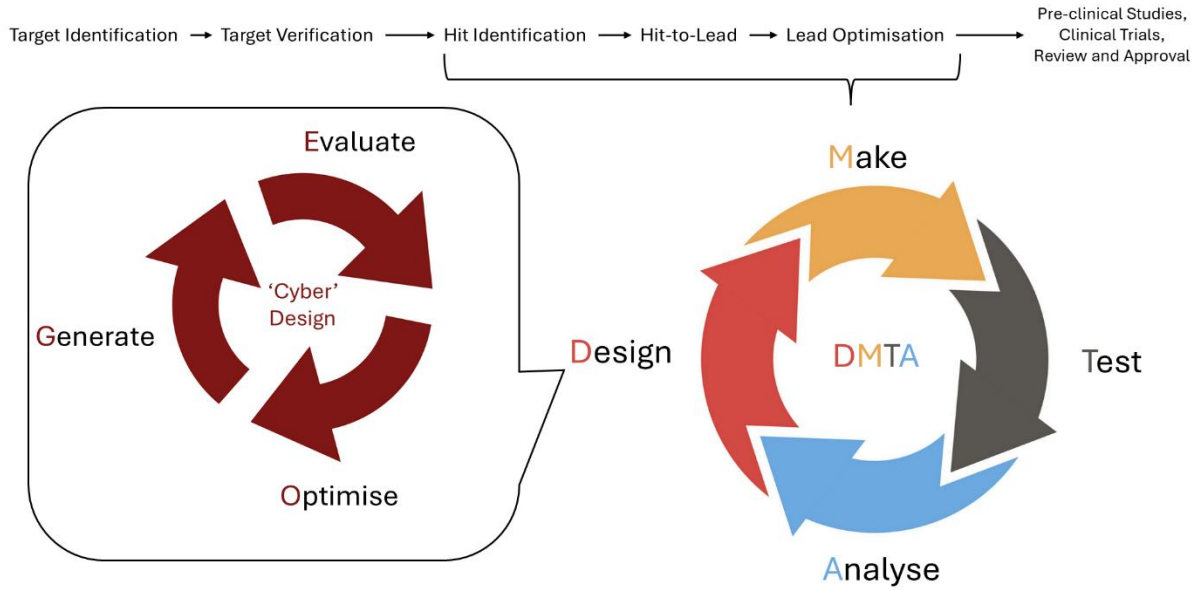


Figure 2-1. The Drug Discovery Pipeline and where DMTA Cycle Fits. The DMTA cycle fits within the drug discovery phase of the pipeline. The design stage of the DMTA can be further broken down into a cycle of Generate, Evaluate, and Optimise in what is known as 'Cyber' Design.

The Design step involves designing or selecting compounds with the potential to interact with a target and modulate its activity. Key questions that need to be addressed in this step include identifying the core components that contribute to a molecule's activity; the diversity of functional groups substituted into the core, when side chains should be used, and etc. (Wesolowski and Brown, 2016). This stage of the DMTA remains one of the major challenges in drug discovery (Ghiandoni *et al.*, 2024).

The Make step involves the synthesis or acquisition of compounds that have been designed or selected during the Design step. If a compound was designed, chemists need to synthesise the compound or its analogues in the laboratory. If a compound was selected from a library, it needs to be acquired from commercial sources or from in-house libraries. The Make stage is typically more resource intensive compared to the Design stage.

The Test step involves conducting experiments to obtain a series measurements of compound properties. Experimental tests can include *in vitro* testing and *in vivo* testing. *in vitro* testing involves biochemical assays which assess a compound's ability to interact with a target, for example, receptor binding and cell-based assays which use cell cultures or cellular systems to assess a compound's effect on disease-related processes. *in vivo* testing involves the use of animal models to evaluate a compound's pharmacokinetic properties, toxicity, and safety.

The Analyse step involves the examination of the results and data obtained from the Test step. This final step is important for informing further DMTA cycles, decision making, and prioritising promising drug candidates. The main activities in this step are identifying patterns, correlations, and the significance of the experimental outcomes. A common procedure at this stage is to construct computational models that find the relationships between chemical structures and their properties or activity. These techniques are known as quantitative structure-property relationship (QSPR) and quantitative structure-activity relationship (QSAR) modelling, respectively. Information from these models helps practitioners to identify structural features that contribute to a compound's properties and activity, which can inform the optimisation of compound properties.

DMTA can go through many iterations before finding potential candidates. This can cost up to 30% of the overall drug discovery program (DiMasi, Grabowski and Hansen, 2016). Therefore, making the DMTA cycle efficient is an important consideration in drug discovery to produce medicines more cheaply and quickly. This can be done in many ways which includes improving the exploration of chemical space, making many more compounds, and finding ways to synthesise molecules more cheaply and efficiently (Warr *et al.*, 2022). The next section will describe how computer-aided drug design (CADD) can help improve the navigation of chemical space.

2.4 Chemical Space

The chemical space is a multidimensional universe that encompasses all possible chemical compounds, including both known and hypothetical. The dimensions of this space are defined by various structural features, properties and activity (Figure 2-2). Fundamentally, chemoinformatic tools are used to navigate this space to identify promising drug candidates (Medina-Franco *et al.*, 2022). However, navigating the entire chemical space is infeasible due to its size.

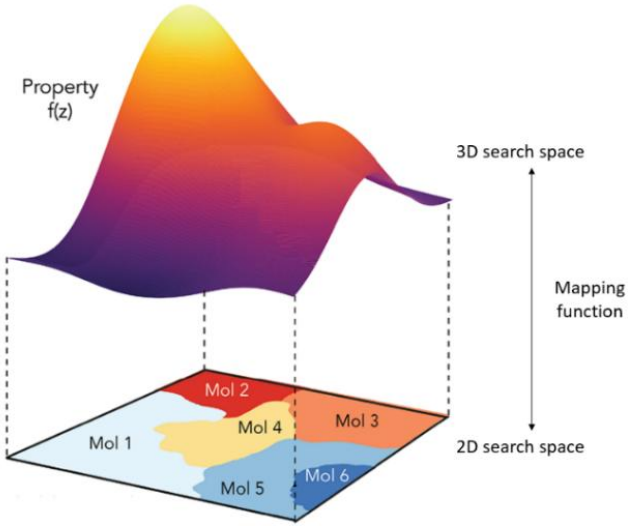


Figure 2-2. A Subset of the Chemical Space in 2D and 3D. The molecules can be mapped into the search space in a discrete manner, i.e. where there are no overlaps between two molecules. In the figure, the molecules are mapped along the x- and y-coordinates of a search space. The addition of additional information such as the property of the molecule increases the dimension of the search space. Adapted from Gómez-Bombarelli et al. (2018).

The currently known chemical space can be split into existing collections and virtual collections (Reymond *et al.*, 2012; Walters, 2019). Existing collections are chemical libraries that contain chemical compounds that have been synthesised and characterised. An example of an existing collection is ChEMBL which contains data extracted from medicinal chemistry literature (Gaulton et al., 2012, 2017; Bento et al., 2014; Mendez et al., 2019).

In contrast, virtual collections consist of hypothetical or computationally generated compounds which are not necessarily drug-like. Examples include the Generated Databases (GDBs) which consist of molecules enumerated up to a given number of atoms of C, N, O, S, and halogens, following valency rules. The Reymond group first published the enumeration of molecules up to 11 atoms GDB-11 (Fink and Raymond, 2007). The most up to date database from the group is GDB-17 (Ruddigkeit *et al.*, 2012). A hybrid of these collections is ZINC (Irwin and Shoichet, 2005), a library of commercially available compounds, that contains both annotated compounds and virtual compounds: the latest version (ZINC22) contains Enamine REAL compounds, a library of enumerated structures (Tingle *et al.*, 2023).

2.4.1 Chemical Space Navigation

Navigating the chemical space is an important aspect of molecular design. In general, the goal of molecular design is to prioritise or generate compounds for experimental testing. Navigation involves exploring the universe of chemical compounds to identify molecules with desired activities. The development of CADD approaches has expedited this process, reducing costs and minimising failures.

Prior knowledge on a drug target of interest can be used to guide the algorithms that are used to navigate the chemical space. This steers algorithms to search areas of the chemical space that are more likely to contain molecules that are desirable for a given drug target. If structural information of the drug target is available, structure-based drug design (SBDD) can be performed. Otherwise, ligand-based drug design (LBDD) may be performed (reviewed in Bassani and Moro, 2023). Both SBDD and LBDD help guide the CADD algorithms in navigating the chemical space.

SBDD uses 3D structural information of a biological target, which can be obtained through X-ray crystallography or NMR spectroscopy, to design molecules. This can be a powerful approach as it can incorporate important information in the design process, such as a compound's docking pose. Examples of SBDD methods include molecular docking, which predicts the ideal docking pose of a ligand within a target's binding site; molecular dynamics, which models the dynamic behaviour of

protein-ligand complexes; and free energy calculations, which predicts the binding affinity of ligands to the target of interest.

In the absence of a target's structural information, known ligands (molecules which bind to a target of interest) are used as a starting point for LBDD. The goal in LBDD is to use existing data to identify patterns and correlations to guide the downstream design of compounds. More specifically, LBDD defines a desirable neighbourhood in the chemical space around known active molecules. For example, a common method is quantitative structure-activity relationship (QSAR). QSAR is a computational approach for predicting the activity or other properties of a compound based on its molecular structure. Structural features of interest may include size, shape, and functional groups.

2.5 Molecular Representation

Several molecular representations have been developed for the computational processing of chemical structures. This section discusses the most popular types of molecular representation for CADD applications including de novo design.

2.5.1 Molecular Graphs

Chemical structures are typically stored and handled by computers in the form of molecular graphs (Leach and Gillet, 2007). A molecular graph is the 2D mathematical representation of a chemical structure and depicts its topology, that is, how the atoms are connected. In the molecular graph representation, the atoms and bonds of a molecule are mapped into sets of nodes (circles) and edges (lines), respectively (Figure 2-3). More formally, a molecular graph is undirected and mathematically defined as a tuple $G = (V, E)$, where V represents the set of nodes that corresponds to the atoms and E represents the set of edges that corresponds to the bonds that link the atoms together and their orders.

For a molecular graph representation to be computer-readable the graph *G* needs to be encoded into matrices (Figure 2-3). For example, a graph can be defined by an adjacency/connectivity matrix which indicates how atoms are connected, a node features matrix (which includes the identity of the atoms and their properties), and an edge features matrix (which includes the bond types) (David *et al.*, 2020). A tensor representation of a molecular graph can also be used which simply combines the adjacency, node, and edge features matrices together (Xia *et al.*, 2019).

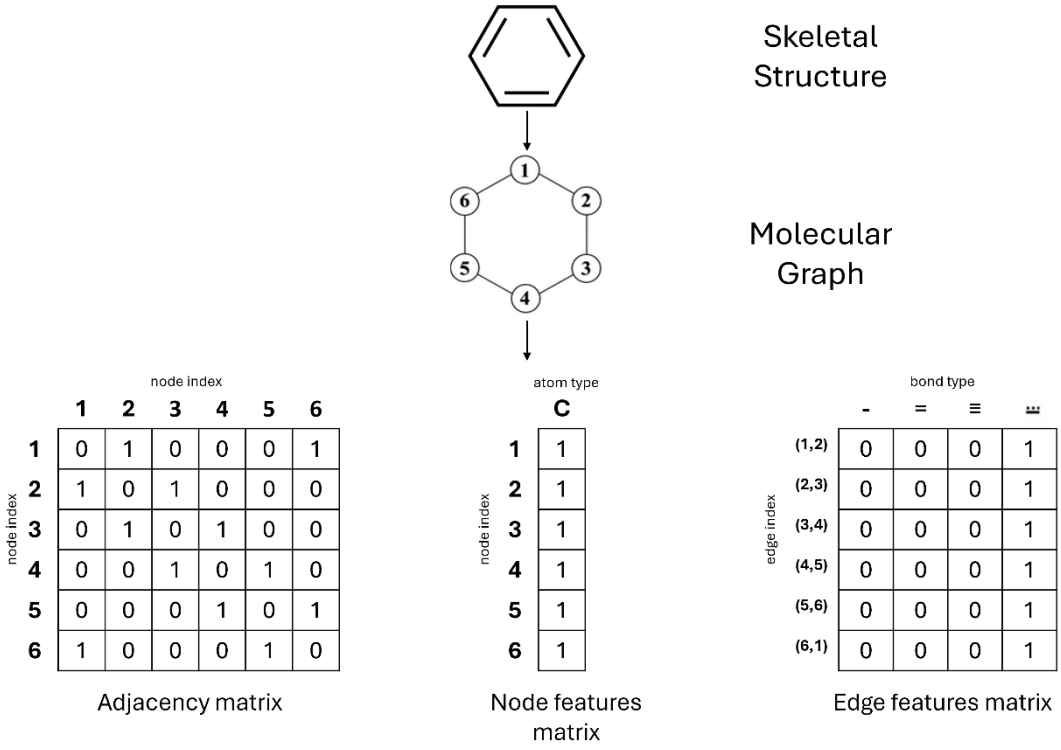


Figure 2-3. Molecular Graph of Benzene Mapped into Matrices. The figure shows the benzene represented in different formats.

2.5.2 Linear Notations/String Representations

Linear notations represent molecular structures as alphanumeric characters, similar to natural language. This makes them compact and more human-readable. Therefore, these representations are preferred when memory and computational power is limited.

Wiswesser Line Notation (WLN)

The Wiswesser Line Notation (WLN) was one of the early and most widely used linear notation (Wiswesser, 1954). However, the WLN requires an extensive knowledge and understanding of its complex rules to generate correct notations. Therefore, it has fallen out of favour as more intuitive linear notations have been proposed, such as SMILES.

Simplified Molecular Input Line Entry System (SMILES)

The Simplified Molecular Input Line Entry System (SMILES) representation (Weininger, 1988) overtook WLN in popularity due to its simplicity and readability (Leach and Gillet, 2007). SMILES uses ASCII (American Standard Code for Information Interchange) characters to represent molecules. The notation conventions are outlined as follows:

- Atoms – represented by their atomic symbols. For example, an aliphatic carbon is represented by the capital letter ‘C’. For two-character atoms, the second letter is entered in lowercase, e.g. Cl. Lower case symbols are used to represent atoms in aromatic rings, e.g. aromatic carbon is represented by the lower case ‘c’. Hydrogen atoms are typically implicit by default.
- Bonds – represented by “–“ for single bond, “=” for double bond, “#” for triple bond, and “:” for aromatic bonds. Single bonds are implicit by default.

- Cyclic structures – represented linearly by first breaking a bond in the ring. The atoms participating in the ring are then enclosed with numbers. For example, benzene is represented as ‘c1ccccc1’, where the 1 at the end closes the ring by connecting the first and last c.
- Branches – represented by enclosing the atoms in parentheses. These can also be nested. For example, 2-methylbutane can be represented as ‘CC(C)CC’.
- Enantiomers - represented by @ where the neighbours are listed counterclockwise and @@ where the neighbours are listed clockwise.

The construction of SMILES requires the traversal of the molecular graph (Figure 2-4). Pre-processing of the molecular graph first involves the trimming of hydrogen nodes from the molecular graph. The cycles in the graph are then broken by removing one edge within each cycle. This turns the molecular graph into a spanning tree, which is a subgraph that includes all the vertices of the original graph but contains no cycles. This means that there is a unique path between any pair of vertices which simplifies graph traversal. Finally, depth-first tree traversal is used to visit and process all the nodes just once in the spanning tree. The depth-first tree traversal algorithm is outlined as follows:

1. A root node (i.e. a starting atom) is designated.

   This serves as the starting point for the traversal. The root node of choice may be random or be based on rules such as a canonicalization algorithm (discussed below).
2. Traversal of the molecular graph.

   Starting from the root node, a depth-first traversal is performed. When a dead end is reached or there is a node with no unvisited neighbours, the algorithm backtracks to the previous node and explores alternative paths until all nodes have been visited.

3. Encoding of the molecular graph.

   As nodes are visited during the traversal, their information is encoded into the SMILES string. If a branching point is encountered, these are indicated using parentheses in the SMILES string. The branch point is traversed until the end, after which the parentheses is completed. For ring structures, numeric labels are used to indicate the connectivity of atoms separated by cycle breaking.

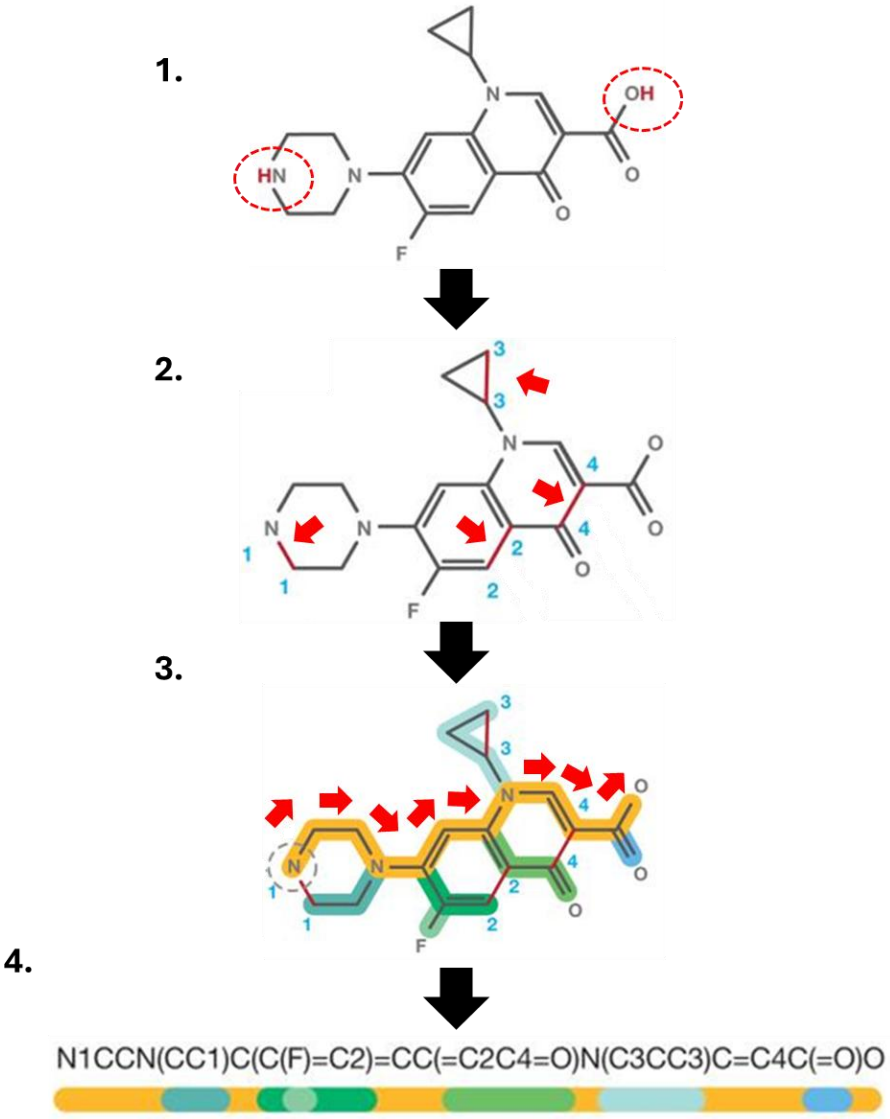


Figure 2-4. Constructing a SMILES String for Ciprofloxacin. The conversion of a two-dimensional chemical graph into a SMILES string. This involves: 1) the removal of hydrogens, 2) the removal of one bond/edge in each cycle, and 3) the traversal of the chemical graph and printing the symbols encountered. Parentheses are used to indicate branchpoints in the tree. Figure adapted from Hodos et al. (2016).

2.5.2.1 Canonicalization algorithms

A classical problem in chemoinformatics is uniquely identifying molecular structures. This process is important for identifying structures from a database and other computational analyses. However, an issue in some molecular representations is that for a given molecular structure there may be more than one valid depiction. For example, the way a SMILES string is constructed means that there can be different SMILES strings that represent the same molecular graph. SMILES can be non-unique because there are different choices that can be made during the construction process, the choice of bonds for cycle breaking, the choice of starting node, and the choice of path when encountering branch points. This leads to a variation in the numbering of atoms which in turn influences the graph traversal process (Figure 2-5).

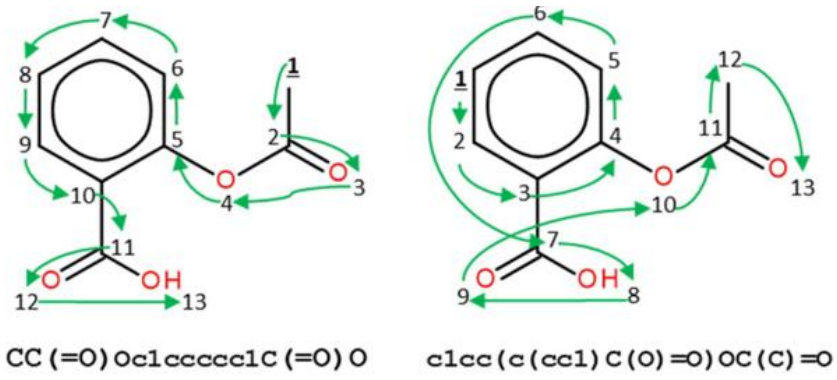


Figure 2-5. Graph Traversal of the Aspirin Molecule. Note how the numbering of the atoms in the Lewis structure affects the graph traversal algorithm. Figure taken from David et al. (2020).

Canonicalisation of a molecule usually consists of two phases: initialisation, where each node is given a value based on their local properties, known as invariant; and relaxation, where the node values are iteratively refined until a unique value for each node is obtained.

Morgan Algorithm

One of the most widely used methods to determine the canonical order of atoms is the Morgan algorithm (Morgan, 1965). The first step is initialisation, where each node is assigned a connectivity value equal the number of atoms it is connected to. The next step is relaxation, where the algorithm iteratively updates the label of each atom based on the labels of its neighbouring atom. This iterative calculation continues until the number of different connectivity values reaches a maximum. Priorities can then be assigned according to the connectivity values. For ties, additional properties may be considered such as atomic number or bond order.

2.5.3 Molecular Descriptors

Molecular descriptors are the quantitative representation of a molecule's properties and are crucial for many chemoinformatics applications. These representations may either be experimentally measured or algorithmically derived. Molecular descriptors can be broadly classified based on the information they encode. The main categories are 1D, 2D, and 3D descriptors.

1D Descriptors

One-dimensional (1D) descriptors are the simplest which represent molecular properties that can be described by a single numerical value or counts. Examples of 1D descriptors are molecular weight, simple counts, for example, hydrogen bond donors and acceptors, and physicochemical properties, such as logP.

2D Descriptors

Two-dimensional (2D) descriptors capture information about molecular topology or connectivity without considering the three-dimensional spatial arrangement of atoms. One of the most commonly used 2D descriptor is the 2D fingerprint. 2D fingerprints are generated using either dictionary- or hashed-based methods.

Dictionary-based fingerprints use a predefined list of common substructures to encode the features of a given molecule in a binary vector, where ‘1’ indicates the presence and ‘0’ indicates the absence of a substructure. The advantage of dictionary-based fingerprints is that each encoded position typically corresponds to a substructure, making them more interpretable. For example, in the Molecular ACCess Systems (MACCS) fingerprints, each bit corresponds to one of the 166 predefined structural fragments determined by medicinal chemists (Durant *et al.*, 2002).

On the other hand, hashed-based fingerprints first break the molecule into smaller parts, such as paths or the circular environment of an atom. Paths are molecular features extracted by analysing the linear sequence of atoms along a molecular graph. In contrast, circular fingerprints capture information about each atom’s neighbouring atoms and bonds within a circular manner. The size of the circular environment is dictated by a radius parameter, where a smaller radius captures more local information, and a larger radius captures more global information by considering atoms that are further away from the central atom. An example of a circular fingerprint is the extended connectivity fingerprint (ECFP) which is calculated using a variant of the Morgan algorithm (Rogers and Hahn, 2010). These dynamically generated substructures are then processed through a hash function to create a unique code called a hash value which is then used to set a bit to ‘1’ in a binary vector.

3D Descriptors

Three-dimensional (3D) descriptors capture information about the spatial arrangement of atoms in a molecule, providing information about its shape. This makes them important for understanding and predicting drug-receptor binding. An example are pharmacophore keys, which represent the spatial arrangement of molecular features such as hydrogen bond donors/acceptors, charged centres, and hydrophobic regions (Schaller *et al.*, 2020).

2.5.4 Virtual Screening

Virtual screening (VS) is a computational method that is used to search, score, rank and/or filter a chemical library using either ligand-based or structure-based methods. The general VS workflow is outlined in Figure 2-6. The result of VS is subsequently used for compound selection to aid processes such as biological screening, compound acquisition, library design, and synthesis (Leach and Gillet, 2007).

The first step is to use general compound filters in an attempt to separate “drug-like” compounds from general organic compounds. The overall aim of which is to improve the number of lead molecules being identified. Filters may include Lipinski’s rule of five (Lipinski and Hopkins, 2004), which ensures good absorption, and pan-assay interference compounds (PAINs) (Baell and Holloway, 2010) which is designed to remove compounds that typically give false positive results in assay readouts.

If the structure of the target is known, protein-ligand docking can be performed. Protein-ligand docking predicts how small molecules bind to a target protein’s active site. Docking algorithms perform this by first generating poses of the ligand within the active site using a search algorithm and then identifying the most likely ligand pose. The simulation and evaluation of the binding interactions then help prioritise compounds from the input chemical library.

If the structure of the target is unknown, ligand-based methods can be performed. Ligand-based methods can be divided into methods that require actives only and those that can be used when both actives and inactives are known. If only actives are known, then similarity searching or pharmacophore mapping can be performed.

Comparing the likeness of molecules is a key concept in both chemoinformatics and medicinal chemistry and is used to search for molecules similar to compounds of interest. The rationale for similarity searching is the Similar Property Principle (SPP), the observation that structurally similar molecules have similar properties and biological activity (Johnson and Maggiora, 1990).

Quantification of molecular similarity requires the following: a representation to encode molecular/chemical features of interest, for example molecular fingerprints (discussed above), and a similarity function/coefficient to process the encoded information within the feature representation.

Many similarity coefficients exist but the Tanimoto Coefficient (Tc) is the most commonly used similarity measure because of its ease of use and speed (Maggiora *et al.*, 2014). Tc is defined in equation (1), where a is the number of features present in molecule A, b is the number of features present in molecule B, and c is the number of features present in both A and B. Tc therefore quantifies the fraction of features present in both A and B compared to the overall number of features, double counting corrected for by - c. So, as A and B become more similar, Tc(A, B) approaches 1.

$$Tc(A,B) = \frac{c}{a+b-c} \quad (1)$$

Pharmacophore mapping identifies the 3D arrangement of features in a set of active compounds that contributes to their biological activity. This method involves extracting essential features from the active ligands, aligning the active compounds in 3D space to identify the spatial relationship between the features, and then generating a pharmacophore model which indicates the spatial relationships between the features (Leach and Gillet, 2007). The pharmacophore model can then be used to identify new compounds that contain it. If both the actives and inactives are known, machine learning (ML) methods can be used. This method uses ML methods to model QSAR/QSPRs and subsequently assess the input search space.

QSAR/QSPR modelling is a technique used to predict the biological activity or chemical properties of molecules based on their chemical structure (Hansch *et al.*, 1962). The general workflow of QSAR modelling is as follows: defining/calculating the molecular features, modelling (i.e. a model maps chemical structure to activity or property), and model validation (e.g. cross validation). Traditional machine-learning methods that has been used include random forest and regression. Recently, the application of deep learning models has been gaining traction, termed 'deep QSAR'. The advantage of this approach and is that deep QSAR is more scalable and able to capture more complex relationships (Gini *et al.*, 2019; Tropsha *et al.*, 2024)

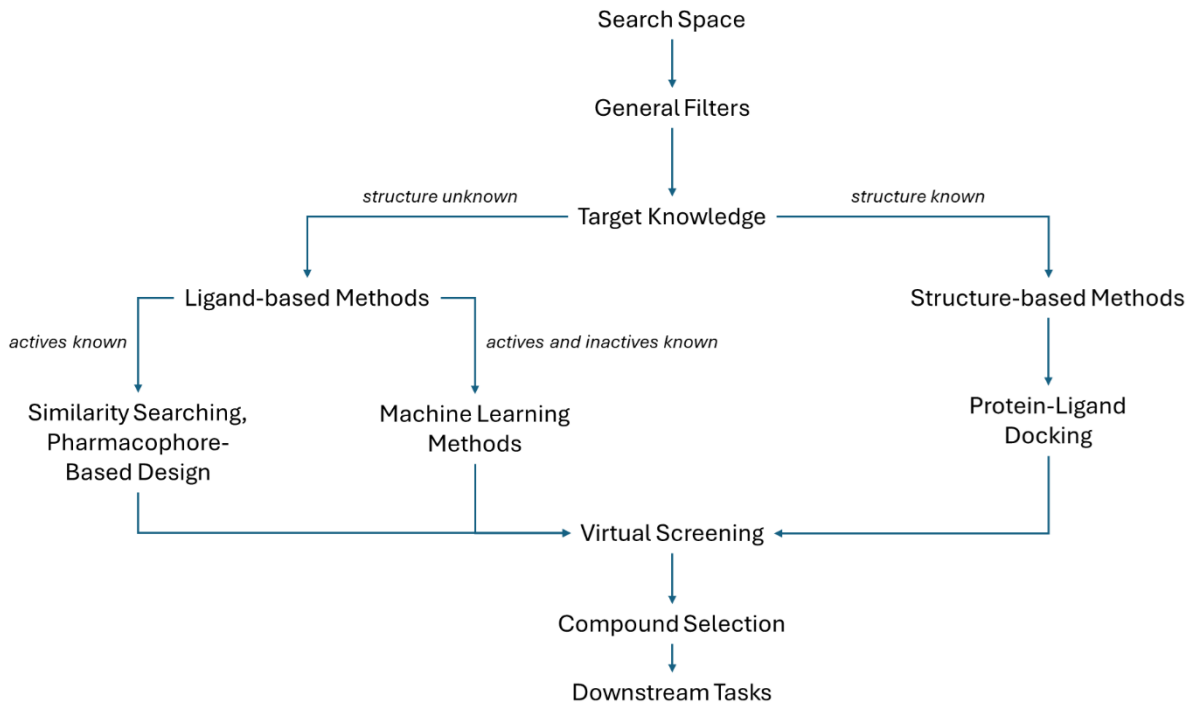


Figure 2-6. General Workflow of Virtual Screening.

2.6 Conclusions

The first section of this chapter explores the different drug discovery stages, from target identification to clinical trials. The next section describes how the early drug discovery stages (hit identification, hit-to-lead, and lead optimisation) is commonly portrayed as a DMTA cycle. Next commons ways of representing molecules in computer-readable form are presented alongside commonly used molecular descriptors. The molecular descriptors form the basis of virtual screening techniques which are used to search databases of existing compounds to identify compounds for experimental testing. The next chapter provides an overview of *de novo* design which refers to the design of novel molecules from scratch.

Chapter 3. De novo Design

3.1 Introduction

De novo design is a computer-aided drug design method that creates compounds from scratch which fit some user-defined property profile. Structures are generated directly by *de novo* design methods. By guiding the generation process with a specific set of design criteria, *de novo* design methods can propose chemical structures that are more likely to be optimal given a set of objectives. This is in contrast to molecular search and virtual screening (VS) where structures are known *a priori*, i.e. are in existing compound libraries. Methods that rely on compound libraries are limited because these libraries only correspond to a small proportion of the entire chemical space. Therefore, they are unable to identify molecules that may provide the best score in a search space (Meyers, Fabian and Brown, 2021). Ideally, *de novo* design will form an *in silico* loop of the Design stage of DMTA, described in Chapter 2 (Thomas *et al.*, 2022). However, a significant challenge remains in ensuring that *de novo* generated molecules are synthesisable.

De novo design methods can generally be broken down into the three core components of assembly, scoring, and optimisation/search. These components work together to rationally design molecules. Assembly dictates how molecules are generated. Scoring involves evaluating generated molecules given a constraint. Optimisation, or search, combines assembly and scoring to dictate how a model should navigate the chemical space towards a user-defined goal. The first *de novo* design programs were introduced in the late 1980s and relied on incorporating rule-based optimisation. More modern approaches incorporate generative deep neural networks.

This chapter first describes the different components that make up *de novo* design. Then it explores the different methods that have been developed, from the traditional to the more modern generative-based models. Finally, benchmarks and the current limitations in *de novo* design are described.

3.2 Assembly/Construction Strate

Assembly dictates the way in which molecules are built from scratch. The assembly strategy can be split into either atom-based, fragment-based, or reaction-based.

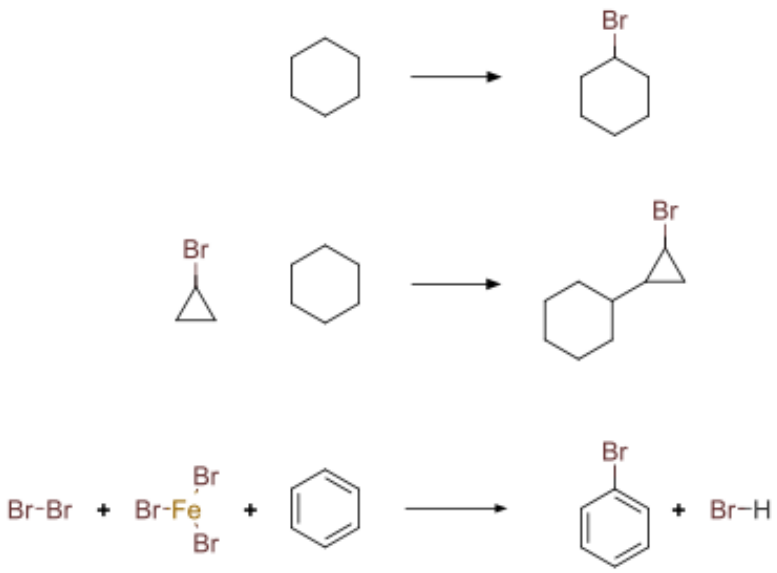


Figure 3-1. Assembly Strategies. From top to bottom, atom-based, fragment-based, and reaction-based. Taken from Webster (2021).

3.2.1 Atom-based Methods

Atom-based methods involve the assembly of molecules atom-by-atom from a starting point, which may be a seed atom, seed fragment, or an empty placeholder. Atom-based assembly may be used to create an entire molecule or link two different molecules. One of the first methods to employ atom-based assembly is CONCEPTS (Creation of Novel Compounds by Evaluation of Particles at Target Sites), which fills a protein's binding site of interest with unconnected atoms, these atoms are then allowed to move within the binding site, finally the atoms are connected (Pearlman and Murcko, 1993). Atom-based methods provide the highest degree of granularity, allowing access to the entire chemical space. However, this comes at the cost of being unable to undertake a systematic approach to searching the chemical space (Hartenfeller and Schneider, 2011).

3.2.2 Fragment-based Methods

Fragment-based approaches use fragments as building blocks to assemble molecules (Hartenfeller and Schneider, 2011). The fragments can include single atoms, substructures, and ring systems. Fragment combination methods can be generally classified into fragment growing (the extension of the molecule of interest via the addition of functional groups) or fragment linking (i.e. the linking of multiple fragments to form one compound) (Sliwoski *et al.*, 2014). Examples of fragment-based *de novo* design include SPROUT, TOPAS/Flux, and SkelGen, reviewed in (Schneider et al., 2010). Briefly, SPROUT was one of the earliest *de novo* design approaches for ligand design. SPROUT involves a two-step process: 1) molecules are built up fragment-by-fragment with dummy atoms based on satisfying steric constraints, 2) elements are then assigned based on electrostatic and hydrophobic constraints. Any remaining undefined atoms are by default set to carbon (Gillet *et al.*, 1994). SkelGen also consists of a two-step process: 1) skeleton generation and 2) element type assignment. This program uses simulated annealing for the optimisation process (Todorov and Dean, 1997). TOPAS/Flux aims to construct molecules that have high similarity to one or more reference ligands. The construction process uses molecular fragments derived from virtual retrosynthetic RECAP rules as building blocks (Schneider *et al.*, 2000).

3.2.3 Reaction-based Methods

Reaction-based methods use reaction transformations to construct molecules. The key advantage of this method over atom-based and fragment-based methods is that the generated molecules are more likely to be synthetically feasible as they follow known chemical transformations (Gillet, Bodkin and Hristozov, 2013). Reaction transformations may include the addition of a single atom, substructure, or ring system. These transformations can be handcrafted or derived from data. For example, one of the earliest methods is SYNOPSIS. The algorithm has 70 possible different reaction types that can be randomly chosen to perform a reaction with a molecule's functional group to generate a new molecule (Vinkers *et al.*, 2003).

3.3 Scoring Strategy

Scoring is the process of assigning a numerical score to a molecule based on various properties of interest. Therefore, this can be imagined as the virtual equivalent of assay testing for activity. How a score is computed can be categorised into either structure-based or ligand-based scoring.

3.3.1 Structure-based Scoring

Structure-based, also known as receptor-based, scoring was the predominant method used in the early development of *de novo* design software. This scoring method evaluates the potential interactions between a molecule and a target's binding site (i.e. the ligand-pocket complementarity), similar to virtual screening. Thus, structure-based scoring requires knowledge of the target of interest's 3D structure. For example, LUDI generated molecules by combining fragments that were then scored based on their potential to form interactions with a protein's binding site, calculated by primarily using hydrogen bonding and hydrophobic contact information (Böhm, 1992).

3.3.2 Ligand-based Scoring

Ligand-based scoring compares a molecule of interest to a reference ligand(s) with desired properties. The method only requires the calculation of molecular descriptors to compare a query and reference molecule(s). The approach is particularly useful when there is a lack of 3D structural information for a target of interest. Additionally, it is less computationally demanding compared to structure-based scoring.

A range of molecular descriptors may be used to calculate a ligand-based score based on molecular similarity to the reference ligand(s), exploiting the similarity property principle (Johnson and Maggiora, 1990). For example, one of the earliest methods to employ ligand-based scoring is the Chemical Genesis algorithm, which calculated multiple molecular properties such as molecular weight (Da) and LogP (Glen and Payne, 1995). Recent developments in machine learning and increasing data have also popularised the use of QSAR models for ligand-based scoring (Cherkasov *et al.*, 2014; Muratov *et al.*, 2020; Tropsha *et al.*, 2024).

3.3.3 Multi-objective Scoring

It is common in drug discovery projects that multiple properties, such as activity, pharmacokinetics, and physicochemical properties, are used to evaluate potential drug candidates. Therefore, in *de novo* drug design, multi-objective scoring is also used to optimise compounds across several dimensions (Gillet, Bodkin and Hristozov, 2013). The two main methods to calculate a multi-objective score are the desirability function and pareto optimisation.

Desirability Function

The desirability function combines different scores into a single score that reflects the overall quality of a molecule (Derringer and Suich, 1980). To calculate a desirability score, desirability functions d_i transform the different properties p_i into a desirability scores between 0 and 1. The transformation may be linear or nonlinear. The overall desirability score D can then be calculated as the geometric mean or weighted sum of the individual desirability scores. For example, the quantitative estimate of

drug-likeness (QED) takes eight physicochemical property descriptors (molecular weight, LogP, H-bond donors/acceptors, charge, aromaticity, stereochemistry, and solubility), transforms them with their own individual desirability functions, and finally combines them by taking the geometric mean of the individual functions (Bickerton *et al.*, 2012). Additional examples are the multi-property optimisation (MPO) tasks in GuacaMol (Brown *et al.*, 2019), described later.

Pareto Optimisation

In Pareto optimization, the goal is to find a set of compounds that represent the best trade-offs between multiple objectives, forming what is called a Pareto front. No single compound on this front is strictly better than another, but each represents an optimal balance of properties where improving one would compromise another. Pareto optimization is commonly used when properties cannot be easily combined into a single score. For example, increasing a compound's potency might decrease its solubility, and Pareto optimization helps identify the compounds that offer the best compromise between these conflicting objectives. For example, the Compound Generator (CoG) program applies a Pareto ranking scheme to evolve the 'median' molecules found between the frontier of a set of molecules (Brown *et al.*, 2004).

3.4 Optimisation/Search Strategy

As mentioned earlier, the chemical search space is vast (in the size of 10^{60}) which makes it impossible to exhaustively explore. To navigate this effectively, an optimisation/search component is required to drive solutions towards those that maximise the given score function. The aim is to iteratively refine the design of molecules to move towards areas with higher fitness scores. The optimisation/search strategies can be categorised into stochastic or deterministic search.

3.4.1 Stochastic

Stochastic search algorithms introduce randomness in the decision process. For a given point in space, the algorithm randomly samples the nearby, i.e. 'local', space to extrapolate information and then subsequently moves in the direction of more desirable space. It is important to note that stochastic search algorithms employ heuristics, so they are not guaranteed to find the global optimum of the chemical space (Hartenfeller and Schneider, 2011). A key example of a stochastic method is the evolutionary algorithm.

Evolutionary algorithms are a family of optimisation techniques that have been inspired by the process of natural selection. These algorithms work by maintaining a population of candidate solutions (molecules) that evolve over time through processes analogous to biological evolution, such as mutation, crossover, and selection. An early example is the evolution of SMILES strings to meet a fitness criterion (Weininger, 1995).

The general workflow of evolutionary algorithms is as follows:

1. Initialisation - A population of candidate molecules is generated randomly or seeded with known molecules
2. Scoring - Each molecule in the population is scored based on one or more functions that measure how well the molecule meets user-defined criteria.
3. Selection - Molecules with higher fitness scores are selected more often to act as "parents" for the next generation.
4. Variation Operators - These operators introduce diversity into the population by generating new candidate solutions from the selected parents.
5. Replacement – After the variation operation, the new molecules (offspring) are evaluated using the fitness function, and the population is updated by replacing some or all of the old molecules with the new ones.
6. Steps 2 to 5 are repeated until the termination condition is met, such as generation number or meeting a fitness score threshold.

Genetic algorithms (GA) are an example of an evolutionary method that use the mutation and crossover operators to introduce variations into the population (step 4 in the workflow above). GAs remains a popular *de novo* design approach as they are simple and remain competitive against generative *de novo* design models. Examples of GA-based *de novo* design include the graph-based GA (Graph-GA) method. Graph-GA performs crossovers and mutations using graph representation rather than string-based representations such as SMILES, which has been the common approach (Jensen, 2019). Another example is AutoGrow4 which combines a GA with docking for structure-based *de novo* design (Spiegel and Durrant, 2020).

3.4.2 Deterministic

Deterministic search algorithms use rule- or function-based approaches in the decision process. This process does not involve the use of randomness. Therefore, given the same of initial conditions, the algorithm always produces the same output. As a result, this is infrequently used in compound optimisation because the ultimate aim of *de novo* design programs is to generate high scoring and/or diverse molecules. Additionally, the chemical search space is rough and using deterministic algorithms for traversal is difficult.

3.5 Generative Deep Learning

Generative models are a more recent method of *de novo* design that offer a data-driven way of navigating the chemical space. To provide a foundation for the next section on generative *de novo* design, this section will first describe the fundamental concepts involved in deep learning and generative deep learning.

3.5.1 Fundamental Architecture

Deep learning (DL) is a subtype of machine learning (ML) which employs artificial neural networks with many layers, hence 'deep', for statistical modelling. How neural networks perform information processing can be loosely made analogous to that of biological neural networks. The advantage of DL over other ML algorithms is its 'superior' ability to find complex patterns and nonlinear relationships in data (Yuan *et al.*, 2017; Gupta *et al.*, 2018; Merk *et al.*, 2018; Segler *et al.*, 2018; Tan *et al.*, 2020). This is particularly useful in drug design as structure and function often does not correlate linearly (Li *et al.*, 2020).

The fundamental building blocks of deep neural networks (DNNs) are the perceptrons, also known as artificial neurons. DNN algorithms are composed of an input layer, a hidden layer, and an output layer (Figure 3-2A). The forward propagation of information through a perceptron is defined by equation (2) where x is the input data, W is the learned weights, b is the bias, and σ is the activation function. The weights are the learnable parameters in the network. The activation function introduces non-linearity

into the model, allowing the network to learn complex patterns. The bias allows the model to fit the data better by allowing the activation function to shift. Increasing the number of nodes in the hidden layer turns the perceptron into a neural network (Figure 3-2B). In turn, adding many hidden layers (at least more than 3) turns the neural network into a DNN (Figure 3-2C) (Li, Zhang and Liu, 2018).

$$y = \sigma(Wx + b) \tag{2}$$

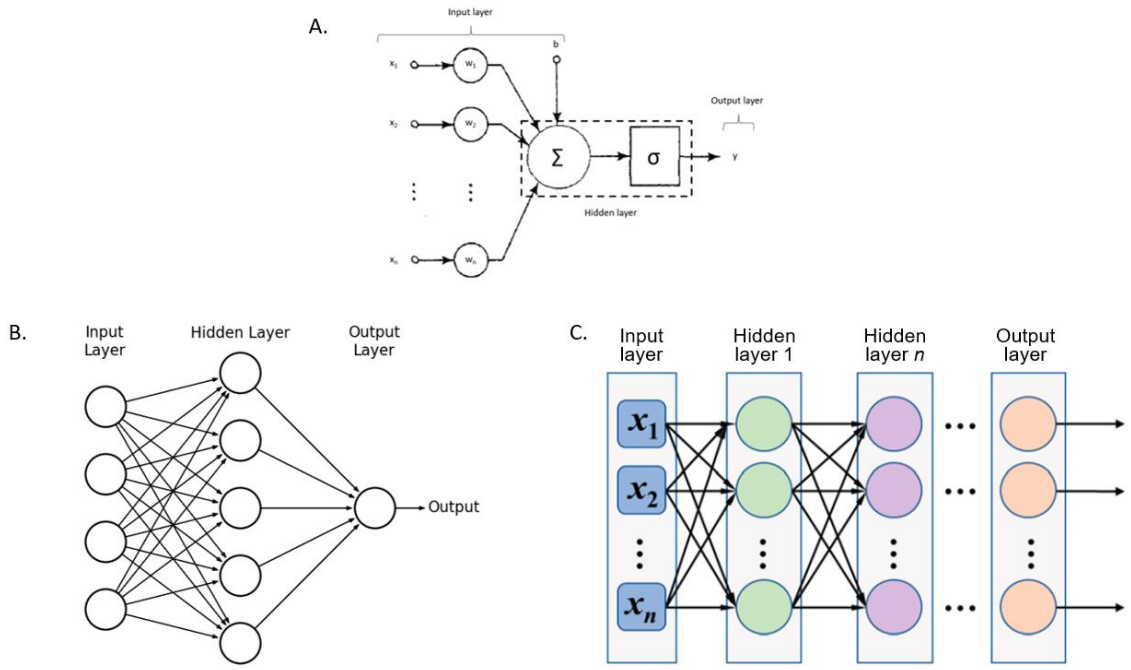


Figure 3-2. Fundamental Architectures of Neural Networks. A, A single layer perceptron. The input layer is composed of x_n, b, and w_n; the hidden layer is composed of a single node which performs matrix multiplication and summation of the inputs from the previous layer (x_n, W_n, and b); and the output layer contains the result of performing an activation function σ on the hidden layer. Figure adapted from Popova et al. (2019). **B,** A neural network. Figure adapted from Olivecrona et al. (2017). **C,** A deep neural network. Figure adapted from Popova, Isayev and Tropsha (2018).

3.5.2 Model Training

Once the architecture is defined, the model needs to be trained to obtain useful outputs. A common way to train DNNs Is using backpropagation which is combined with a gradient-based optimisation algorithm, such as stochastic gradient descent. The first step is to perform a forward pass through the network, i.e. the input data is passed through from the input to output layer. At the output layer, the loss function compares the predicted output with the true label and a loss value is calculated. Backpropagation then calculates the gradient of the loss function with respect to the weights in the network. Once the gradients are computed, the weights and biases are updated using an optimisation algorithm, the most commonly used being stochastic gradient descent.

3.5.3 Deep Learning Tasks

Deep learning is typically divided into discriminative and generative models. Both discriminative and generative DL models have seen increasing application in drug discovery, for example, quantitative-structure activity relationship (QSAR) modelling and generative de novo design, respectively.

Discriminative Model

Discriminative models draw decision boundaries between the different classes in the data space. The aim here is to model the decision boundary by estimating the conditional probability $P(y|x)$ without modelling the distribution of the features themselves, where x represents the input features and y represents target labels. In chemoinformatics, discriminative DNNs are used for property prediction. Here, the task is to perform QSAR/QSPR modelling, whereby molecular features are mapped to the specific properties of molecules, such as solubility and activity.

Generative Model

Generative models aim to estimate the joint probability distribution P(x,y) over the input features x and the outputs y. P(x,y) indicates the likelihood of observing a particular input x and output y together. By modelling the full data distribution, generative models can generate new data instances. In chemoinformatics, generative DNNs are used for *de novo* design. Here, the aim is to generate novel compounds that satisfy user-defined constraints, discussed above.

3.6 Generative De novo Design – Workflow

Generative models were first developed in the computer science domain and have recently been adapted to *de novo* drug design. Traditional *de novo* design methods explicitly model chemical knowledge into the algorithms, i.e. heuristics. However, this is time-consuming, can cause bias, and restrict the ability of *de novo* design programs to efficiently search large areas of chemical space (Gómez-Bombarelli et al., 2018; Kumar et al., 2018; Schneider et al., 2020; Vanhaelen et al., 2020; Yang and Specht, 2019). In contrast, generative models are more direct and generate structures by sampling from a learned implicit model of the chemical space, known as the latent space. These potential benefits may explain the increasing interest in generative model-based *de novo* design (Meyers, Fabian and Brown, 2021).

The general workflow of generative *de novo* design models from input to output is discussed and outlined below (Figure 3-3). This section will be focused on the use of SMILES representation for generative *de* novo design as this is the representation used in the subsequent experimental chapters.

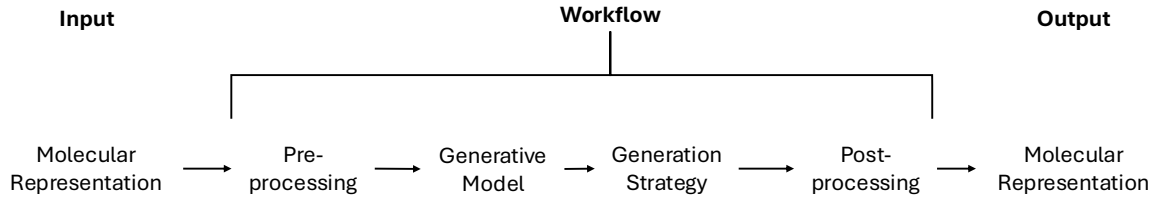


Figure 3-3. The General End-to-End Workflow of Generative de novo Design Models.

3.6.1 Pre- and Post-processing of Text-based Representation

Before text-based molecular representations can be used as inputs into generative models, they need to be converted into tokenised representations. Likewise, the reverse process occurs after generative modelling to output a molecular representation. The section provides an overview on how text data is handled in language modelling.

3.6.1.1 Tokenization

In NLP, tokenization is the process of splitting text data into smaller individual parts, known as tokens. All possible tokens from a given dataset can then be used to define a token vocabulary. Additionally, further preprocessing steps may be performed based on the task. Additional special tokens may be added to the data as required, such as START, END, and PADDING. START and END tokens are crucial for instructing the model when to start and end generation. Whereas PAD tokens ensure that the sequence length is the same for all input, which is required by language models. In NLP, common tokenization methods include the character-, subword-, or word-level tokenization. In chemoinformatics, there are several methods for SMILES tokenization which includes atom-wise tokenization, atom-in-SMILES tokenization, and SMILES pair encoding.

Atom-wise tokenization breaks SMILES to its individual atom characters, numbers, and symbols, as shown in (Figure 3-4). This tokenization is used in the RNN-LSTM from the GuacaMol benchmark for *de novo* design, with the additional processing of double charactered elements into a single token, for example 'Br' is converted into the 'Y' token (Brown *et al.*, 2019). This allows for easier handling and processing of the tokens during the modelling and generation processes.

Atom-in-SMILES (AIS) tokenization introduces environmental information into the tokenization process. Instead of treating all atoms of the same type as identical, AIS considers the local chemical environment of each atom. This is achieved by creating circular atom-centred topological molecular fragments which are defined by the covalent bond radii around each atom (Ucak, Ashyrmamatov and Lee, 2023).

SMILES pair encoding (SPE) creates a vocabulary of high-frequency SMILES substrings. This ensures that the most common SMILES substrings, which often represent significant molecular substructures, are treated as unique tokens (Li and Fourches, 2021). SPE is inspired by the Byte Pair Encoding algorithm, which is commonly used in NLP for subword tokenization (Sennrich, Haddow and Birch, 2016).

3.6.1.2 Vectorization

Vectorization is the process of converting the tokens into a numerical representation. Commonly used SMILES vectorization methods include one-hot encoding and word embeddings. These vectors can then be fed to various types of neural network architectures.

One-hot Encoding

In one-hot encoding, token sequences are represented in a numerical format using a matrix representation. A one-hot encoded matrix is a sparse binary vector, where the rows correspond to the vocabulary and the columns represent each token given a sequence, or vice versa. Figure 3-4 illustrates the one-hot encoding based on a vocabulary size of 2 (representing an aromatic carbon atom 'c' and the ring closure symbol '1'). Every token in the SMILES string is converted into a binary vector, where only the corresponding row index will have a value of 1 and the rest are 0s. The length of the binary vector is equal to the length of the token vocabulary.

Word/Token Embeddings

The Word2vec algorithm is a word/token embeddings that map each token in the vocabulary into a dense, low-dimensional vector in continuous space. The vector representation is generated by training a neural network on a large text dataset. The aim of this model is to capture the semantic relationship between tokens, so that the vector values of closely related tokens are similar. Once trained, an embedding layer creates an efficient mapping of tokens into meaningful vector representations. The dimension of the embedding vector is user-defined.

In chemoinformatics, Mol2vec is an example of a model that learns vector representations of molecules (Jaeger, Fulle and Turk, 2018). Here, substructures derived from the Morgan algorithm are used as “words” and compounds as “sentences”. By applying theWord2vec algorithm, Mol2vec creates vectors where chemical related substructures occupy similar regions in vector space. Additionally, using the Mol2vec vector embeddings as features for supervised ML tasks such as classification and regression have been shown to improve performance compared to Morgan fingerprints.

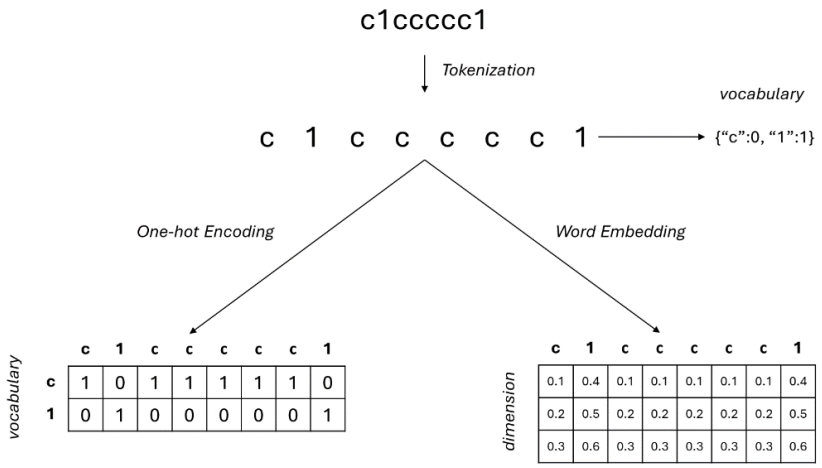


Figure 3-4. Preprocessing of Benzene’s SMILES Representation. The preprocessing step first involves the tokenization of the SMILES string. A vocabulary can then be extracted which is used in the vectorization stage. In the one-hot encoding matrix the columns represent the individual characters of the benzene SMILES strings whilst the rows represent the extracted SMILES vocabulary. In the word embedding matrix, tokens are represented by a learned vector representation.

3.6.2 Generation Strategies

The generation strategy dictates how the learned latent representation is used by the generative model to output a vector molecule representation (which is subsequently post-processed to output a molecule). The two main strategies can be referred to as one-shot and sequential generation processes.

3.6.2.1 One-shot

The one-shot generation strategy outputs the complete molecular structure in one forward pass of the model. This strategy is used by generative adversarial networks and autoencoders, discussed below. When training these models, the full vector representation of the given SMILES is used as input. During the generation process, the full vector representation is decoded to a generated SMILES. A limitation of this strategy is that it is difficult to accurately generate realistic molecules (Du *et al.*, 2024).

3.6.2.2 Sequential

The sequential generation strategy builds the molecular structure step-by-step, either with atoms or fragments. This means that the latent representation is accessed iteratively to inform how a building block is added onto the molecule based on previous building blocks that have already been added. This strategy is used by the chemical language models, recurrent neural networks and transformers, discussed below. Different sequential strategies will impact the quality, diversity, and coherence of generated text. Example strategies include greedy decoding, sampling, and beam search.

Greedy

Greedy decoding is a simple heuristic where the token with the highest probability at each timestep, i.e. after each building block, is selected. This process is repeated until an <EOS> token is reached, or the maximum length is reached. This strategy is typically used when training a language model.

Sampling

Sampling is a method that introduces randomness by sampling tokens from the probability distribution instead of picking the one with the highest probability. The simplest sampling method is to randomly sample a token from the probability distribution at each time step. This random sampling can be controlled using a temperature parameter T, such that when T < 1 the sampling becomes more deterministic and rare tokens are suppressed. In contrast, when T > 1, the sampling tends towards random selection. Sampling allows for more exploration and diversity in the generated output. This approach was first investigated by Gupta et al. 2018b).

Beam Search

Beam search is a method that keeps track of multiple possible sequences at each step. The number of sequences tracked at each step is termed the beam width, top-*b*. The sequences kept are the samples with the highest probability at each step until an end-of-sequence <EOS> token is generated. Figure 3-5 shows an example where the top-2 sequences at each generation step are kept. At each step, the sequence is expanded with the most probably tokens. The top-2 sequences are then kept and expanded further. This repeats until an <EOS> token is reached. The final generation step then outputs the sequence with the top highest cumulative probability. In this example, the SMILES 'CC' has the highest cumulative probability and is therefore generated. This approach was first adopted by Guo and Schwaller (2023).

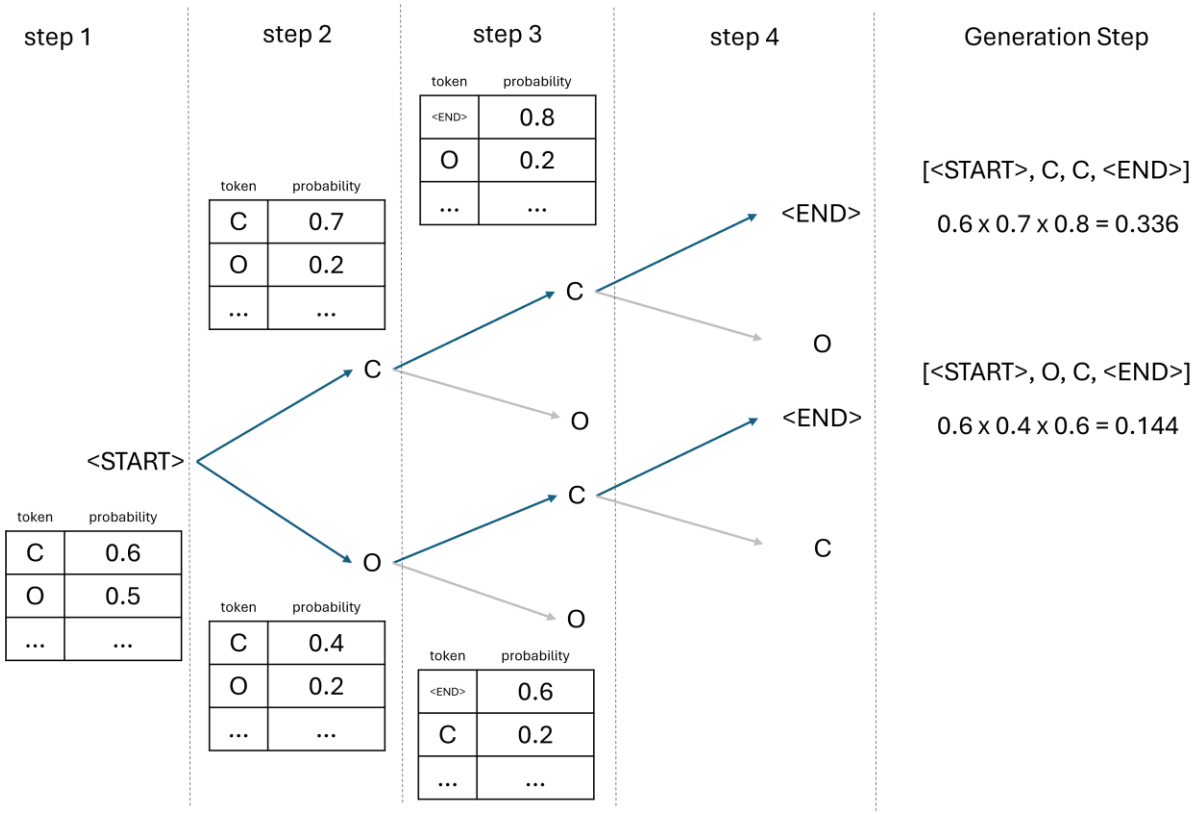


Figure 3-5. Beam Search with Top-2 Beams. The table at each step represents the probability distribution of the next token to be generated given the current token at each step. Beam search keeps track of the two sequences with the highest probability at a given step (indicated by the blue arrows). This repeats until the <END> token is generated. The final generation step then calculates the cumulative probability of the top sequences and returns the sequence with the highest cumulative probability.

3.6.3 Design Tasks

Generative models for de novo design have two design tasks: distribution learning and goal-directed generation.

3.6.3.1 Distribution Learning

Distribution learning is where the model aims to learn the probability distribution of the underlying input data. The probability distribution learnt is the SMILES grammar of the chemical space. To learn this distribution, the model regenerates the input training data during training. With this learned distribution, the model can be sampled to generate novel molecules that mimic the input training data. Here, the only information the model can leverage is the molecular structure. However, the model is still able to follow the property distribution as the training dataset, such as activity, because of the SPP. Distribution learning is also used as a pretraining method for downstream tasks. A pretrained model can be adapted with further retraining. These methods include transfer learning, iterative finetuning, and reinforcement learning.

Transfer Learning/Finetuning

Transfer learning takes advantage of what a model has learnt in one context (a source/prior data) and applies it to another context (a target data) (Figure 3-6A). This process is a single-shot process.

For example, in image classification tasks, models can be pre-trained on the large MNIST (Modified National Institute of Standards and Technology) dataset (source data) - a database containing images of handwritten digits, from 0 to 1. The classifier can then be retrained on a smaller (target) dataset, such as images of dog breeds. Here it is assumed that the learned features from the primary data contain useful information that can help reduce computational cost and help improve performance when retraining on a secondary data.

In the context of *de novo* design, a model may initially be trained on a large dataset such as ChEMBL to learn the general grammar of SMILES. The pretrained model can then be retrained on a set of known active drug molecules, which is typically a much smaller dataset. This retraining allows the model to learn to generate molecular structures that are similar to the known active drugs, whilst retaining the learnt grammar from the initial larger dataset. For example, one of the first methods to use transfer learning initially pretrained an RNN on 550,000 SMILES from the ChEMBL dataset before being finetuned on known 4367 PPARγ inhibitors. The aim of which was to first learn the SMILES grammar before being tasked to generate molecules from a more desirable chemical space (Gupta *et al.*, 2018). Therefore, the pretrained model shifts from a general chemical space distribution into a task-specific application.

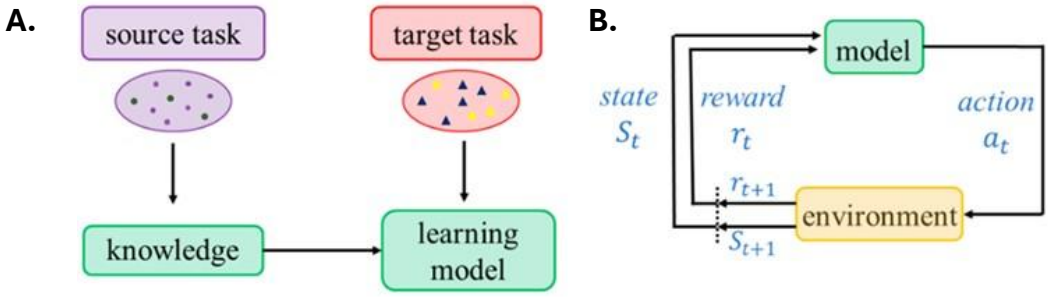


Figure 3-6. Learning Tasks in Machine Learning Models. A, Transfer learning involves using the knowledge gained from a source task and transferring it to a different but related task to improve performance. **B,** Reinforcement learning is concerned with dictating how an agent should act in an environment to maximise a predefined reward. Figure adapted from Yang et al., (2019).

3.6.3.2 Goal-Directed Generation

Goal-directed generation, or molecule optimisation, is a targeted approach whereby the aim is to guide generative *de novo* design models to create molecules that satisfy a given score. Here, the generated molecules are scored according to a user-defined scoring function, and the aim of the model is to improve the score using an optimisation algorithm. Goal-directed generation can be performed with either iterative finetuning or reinforcement learning optimisation methods.

Iterative Finetuning

Iterative finetuning can be performed with a feedback loop guided by some user-defined scoring function. This optimisation/search algorithm is known as iterative finetuning/hill-climbing and was first describe in Neil et al. (2018). Pseudocode is shown below:

```
1. For n-iterations do
2.     for m-sequences do
3.         Sample SMILES from model
4.         Calculate scores of the SMILES
5.     end for
6.     Store generated SMILES in an archive
7.     Take the top-k SMILES with the highest score from the archive
8.     Fine-tune the model on the top-k SMILES structure for n-epochs
9. end for
```

In the pseudocode, the n-iterations determines the number of fine-tuning loops for a given model. The m-sequences parameter dictates how many SMILES are sampled from the model at a given iteration. Finally, the top-k SMILES are used to finetune the model. Iterative/fine tuning can be considered as a form of transfer learning whereby the top-k molecules in each loop are used to retrain the model. Refer to Neil et al. (2018) for more details of the pseudocode and the original implementation. As in transfer learning, the model learns the probability distribution of the input molecules' structure.

Reinforcement Learning

Reinforcement learning (RL) is a biologically inspired framework that attempts to mimic how learning occurs in animals, e.g. using positive reinforcement for good behaviour and punishment for bad behaviour (Figure 3-6B). The main components of an RL an algorithm are:

- Agent – the entity that interacts with the environment and perform actions. Its decisions are based on the current state of the environment. The decision it makes are learned from a feedback loop based on reward and punishment.
- Environment – the external system in which the agent interacts. This includes everything that the agent has no direct control of.
- State – a specific configuration or situation of the environment at a given timepoint.
- Action – the decisions that an agent can perform. This is influenced by the current state of the environment and the learned policy.
- Reward – feedback to the agent indicating the desirability of particular action in a given state of the environment.

More specifically defined, RL guides how an agent learns to interact with its environment based on previous experience using the Markov decision process. This involves rewarding and penalizing the agent based on its action in an environment. The goal of the agent is then to perform actions that will yield the most reward.

In the context of *de novo* design, the agent is the generative model, the environment is the chemical space, the action is a navigation step the agent makes in the chemical space (i.e. adding an atom to a fragment or empty seed, or terminating generation), state is the partial molecule, and the reward is the score of the complete molecule. This task is also iterative, where results from previous iterations are used to retrain the model. However, in contrast to iterative finetuning, RL is able to learn the probability distributions of both the molecular structure and the scoring function. Additionally, it is possible to train the agent from scratch. However, it is common practice to use a pretrained model as the agent because converging to optimal solutions will be faster.

3.7 Generative De novo Design - Architectures

Multiple different generative models have been developed to construct chemical structures for *de novo* design. The different steps in the discussed workflow are typically modular, i.e. different model architectures can use the same input pre-processing methods, training methods, and generation tasks. What separates the different model architectures is the way in which they learn the probability distribution of the input training data. This section describes the different model architectures, such as generative adversarial networks, autoencoders, and chemical language models.

3.7.1 Generative Adversarial Networks

Generative adversarial networks (GANs) consist of two neural networks which are known as the generator and the discriminator. The generator takes random noise z as input and transforms it into an output that attempts to fool the discriminator. On the other hand, the discriminator takes as input both the generator's output and a sample from the training dataset. The discriminator then attempts to identify which data is fabricated and which is real. During training, the generator seeks to improve its ability to fool the discriminator, whereas the discriminator seeks to improve its ability to discriminate between inputs (Liu *et al.*, 2019). The generator and discriminator are trained in an alternate manner until no further improvements are made. The training process is a two-person min-

max game where the objective function is to perform gradient descent on the generator and gradient ascent for the discriminator.

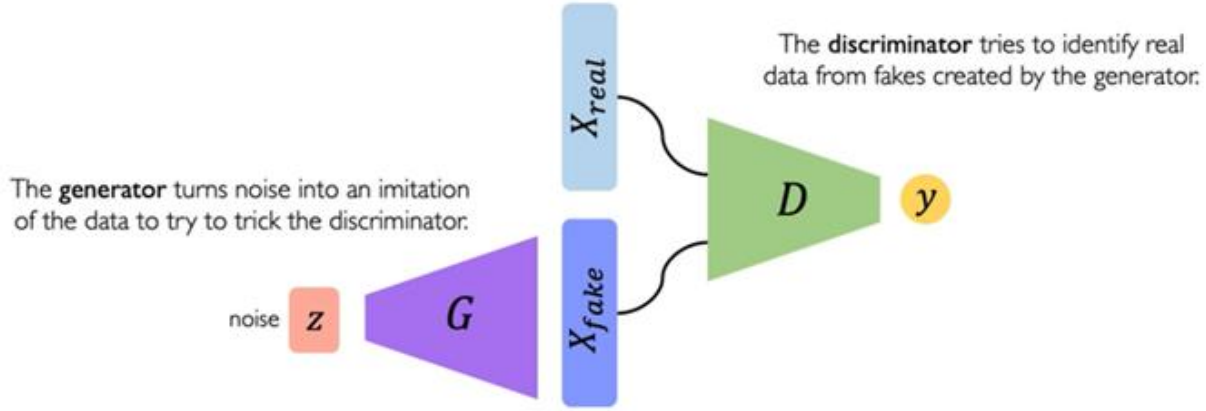


Figure 3-7. Architecture of a Generative Adversarial Network. Generative adversarial network architecture. Figure taken directly from MIT6.S191 (2021).

The successful application of the original GAN model to de novo design has been difficult. The challenge of its application has been likely due to the well-known issue of mode collapse - where the generator is trapped at only producing a small set of molecules since the GAN model has learnt to exploit outputs that are most plausible (Brown *et al.*, 2019). This problem may explain why variants of the GAN (discussed below) were developed for *de novo* design. However, a recent suggested solution for GAN's mode collapse in de novo design was the use of adaptive training data. In Blanchard et al. (2021), the authors borrowed the ideas of replacement and recombination from genetic algorithms to adapt the training data used in GANs. In replacement, highly scoring generated molecules from the GAN are fed back into the training data and have been suggested to prevent training collapse. Whereas in recombination, a subset of the generated molecules is recombined with a sample from the training data. Through replacement and recombination, the authors have suggested that the issue of mode collapse was solved as there was an increase in new molecules that also tend to be high performers.

3.7.2 Autoencoders

Autoencoders (AEs) learn a lower-dimensional feature representation of a given dataset through a self-encoding process. AEs have two main components: an encoder and the decoder (Figure 3-8A). Encoders compress the input data x into a smaller latent representation z to learn important features of the given data in the form of an encoded vector of latent variables, called the deterministic bottleneck. The decoder then attempts to reconstruct x - by sampling the compressed information in z to an output $\hat{x}$. During training the encoder and decoder components are jointly trained. The aim of which is for the encoder and decoders to work together to efficiently compress the data into a lower dimension by minimising the reconstruction loss between x and $\hat{x}$. During training, the aim is to regenerate the input training data. Once the AE has been trained, the latent space can be sampled to decode novel molecular structures.

Variational autoencoders (VAEs) are architecturally similar to AEs but have different mathematical derivations (Ståhl *et al.*, 2019). The original autoencoder map the input x into fixed point vectors z in the latent space. This can leave empty gaps in the latent space and sampling from these gaps can generate invalid outputs. In contrast, variational autoencoders map inputs into a continuous distribution which is achieved by replacing the deterministic bottleneck with a probabilistic operation. The input is now encoded as the mean μ and variance σ^2 vectors of the distribution (Figure 3-8B). By forcing the model to encode the input as a distribution rather than a single point, the latent space becomes smooth and continuous, preventing empty gaps.

Training is maximum likelihood-based to find values for the mean μ and variance σ^2 that best describes the dataset. Additionally, the probabilistic nature of VAEs introduces noise that is a form of regularization. This explicit regularisation limits overfitting and causes the VAE to learn more robust representations during training.

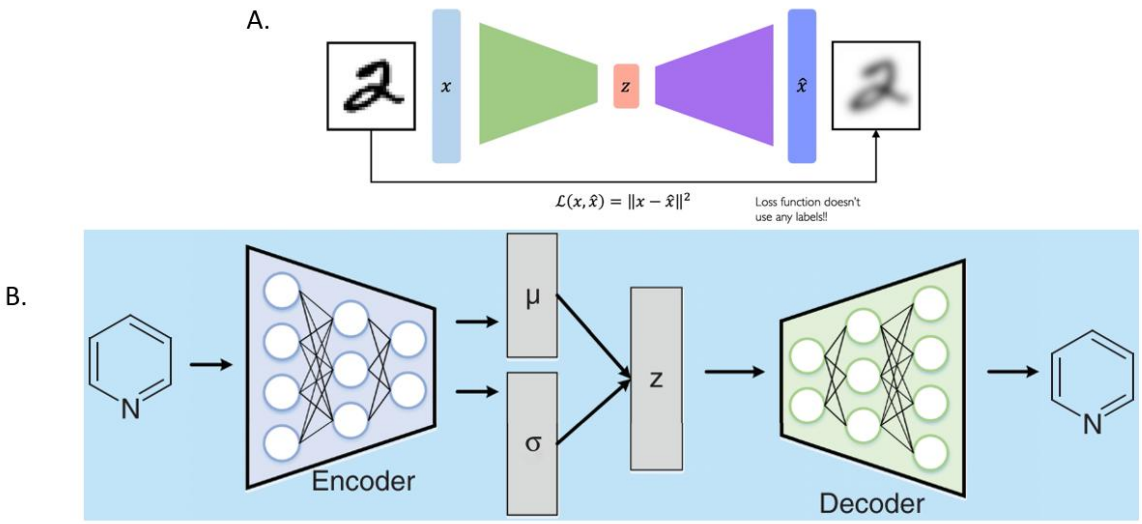


Figure 3-8. Architecture of Autoencoders. A, A traditional autoencoder learning to reconstruct the input digit 2. Figure taken from MIT6.S191 (2021). **B,** A variational autoencoder learning to reconstruct a given molecule. Figure taken from Xu et al. (2019).

One of the earliest studies on VAEs is the work from Gómez-Bombarelli et al. (2018). Here, molecular properties, QED and SAscore, were incorporated in the latent space z during training. Linking the latent representations of molecules to their chemical properties organises the space. This makes it easier to identify regions that correspond to desirable chemical characteristics.

3.7.3 Chemical Language Models

Chemical language models (CLM) represent the intersection between chemoinformatics and NLP. Language models involve the statistical and probabilistic analysis of texts. They involve various tasks such as machine translation, text generation, sentiment analysis, and so on. The language models initially developed for NLP tasks have now been adapted for the handling of text-based molecular representation. These CLMs have seen applications in a broad range of molecule-related tasks such as property prediction, molecule generation and optimisation, and reaction prediction and retrosynthesis. This section focuses on how CLM is applied to *de novo* design (reviewed in Grisoni and Schneider, 2022; Grisoni, 2023).

The application of language models in chemistry, called chemical language models, is based on the idea that chemical structures can be treated as a form of language, an example being SMILES which was discussed earlier. Like natural language, the chemical language also has syntax, where the arrangement of atoms and bonds need to follow specific rules to make chemically valid molecules, and semantics, whereby the way atoms and bonds are arranged leads to different molecular properties (Figure 3-9) (Grisoni, 2023).

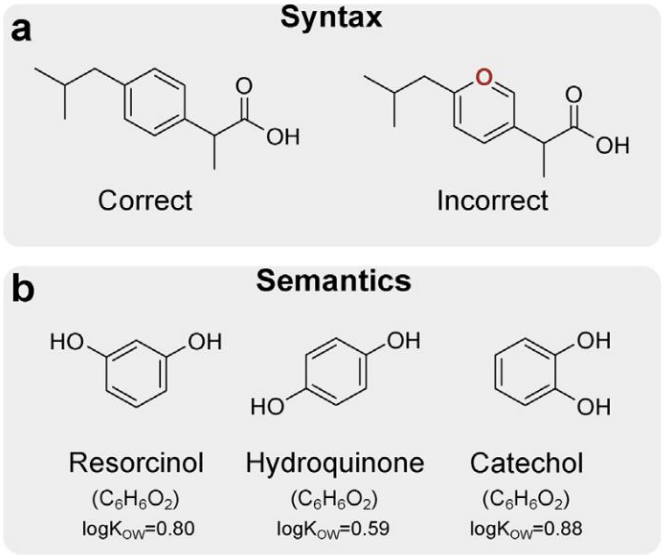


Figure 3-9. The Chemical Language. Like natural language, the chemical language has syntax and semantics. Adapted from Grisoni (2023).

Language models are trained to predict the next value in a sequence based on previous values. This allows the model to learn the patterns and structure found in natural language. However, traditional feedforward networks do not perform well when handling sequential data since sequential data can contain dependencies, i.e., an output may be dependent upon an earlier time step. For example, consider Figure 3-10, where a model is used to predict the next word in the sentence "I love generative *de novo* drug ___". In traditional feedforward networks, the model only remembers information about the current timestep it is in (i.e., "drug" at timestep 6). Consequently, gives similar probabilities to all the possible words after 'drug' and does not give a high probability to the word 'design'. However, since the preceding words before drug are "generative *de novo*", the only correct output should be 'design'.

More modern language models, such as recurrent neural networks and transformers, can remember important information about previous timesteps. This memory allows the model to narrow down its prediction, e.g., giving it a high probability to output the word 'design' following the phrase "generative *de novo* drug" (Figure 3-10). This illustrates the importance for a model to be able to track the dependencies in a sequence, i.e. important contextual information about the sentence.

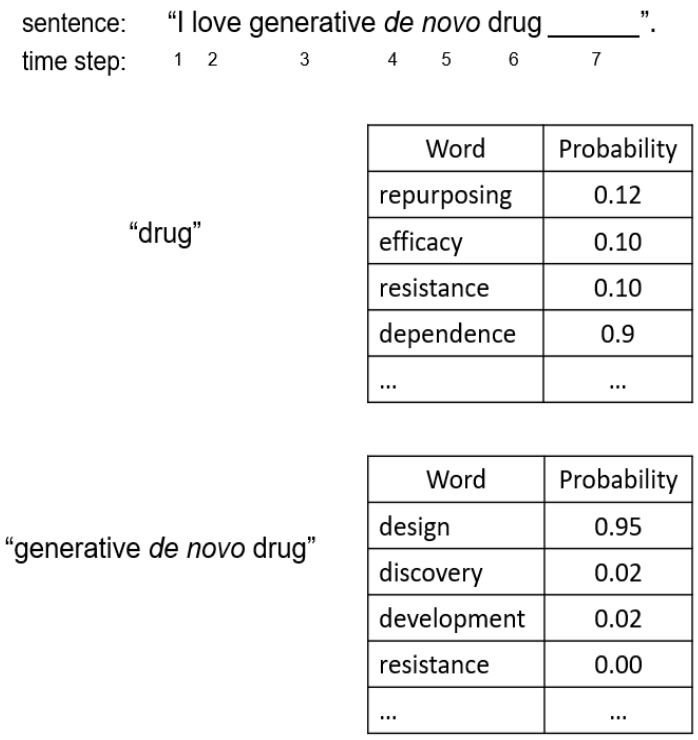


Figure 3-10. Next Word Prediction Using Traditional Feedforward Networks vs Modern Language Models. Traditional feedforward networks only remember the immediate input and thus outputs a similar probability distribution to multiple words. In contrast, modern language models can remember distant inputs and thus the probability distribution becomes more specific and outputs the correct missing word.

To put language models in the context of *de novo* design, consider Figure 3-11. Once a language model has been trained to recreate the input training dataset, the generation process can be performed. Here, the model is fed the initial input of a <START> token. To obtain novel SMILES, the output probability can be sampled as discussed in the sequential decoding strategies section. The sampled token is then fed back to the model, and the sampling continues until an <END> token is reached. In the figure, the SMILES string for benzene has been generated.

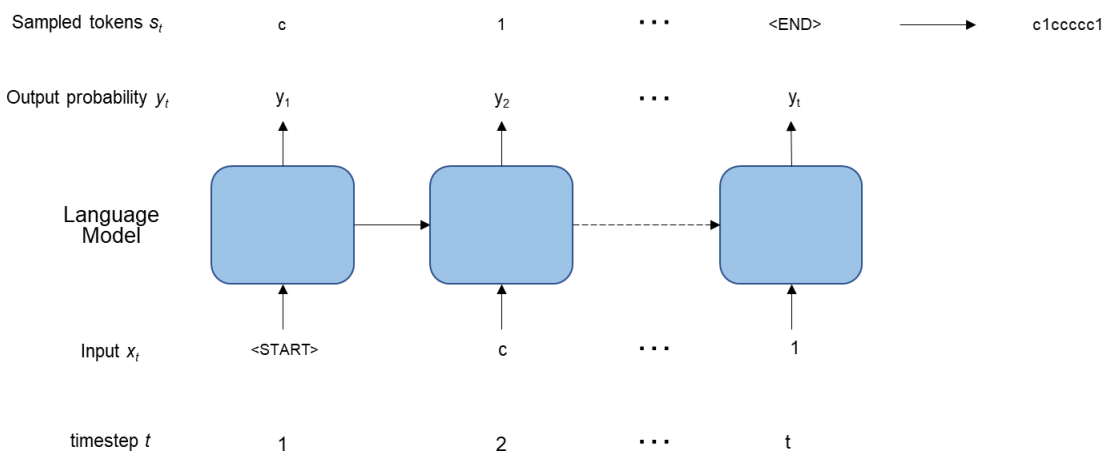


Figure 3-11. SMILES Generation Process of Language Models.

3.7.3.1 Recurrent Neural Networks

The first conceptualisation of recurrent neural networks (RNNs) was from the work of Rumelhart, Hinton and Williams (1986), where they modelled the human brain's ability to process sequential information by introducing internal memory states in artificial neural networks. This feedback loop allows information to propagate from one timestep to another, i.e. previous steps in the sequence are used for predicting the next timesteps. As a result, RNNs are able to handle sequential data and their dependencies.

3.7.3.1.1 Architecture

RNNs use an internal/cell memory state to handle dependencies. This allows important information to be passed on at every time step. So now the prediction becomes dependent upon the immediate input and the cell state. The cell memory state is illustrated in Figure 3-12 and is defined mathematically in Equations 3 and 4, where σ is an activation function; U, W, and V are weight parameters for the input, hidden, and output layers respectively; h_t is the cell state of the RNN which is updated at every timestep t.

$$y_t = \sigma(Vx_t + h_t) \tag{3}$$

$$h_t = \sigma(Ux_t + Wh_{t-1}) \tag{4}$$

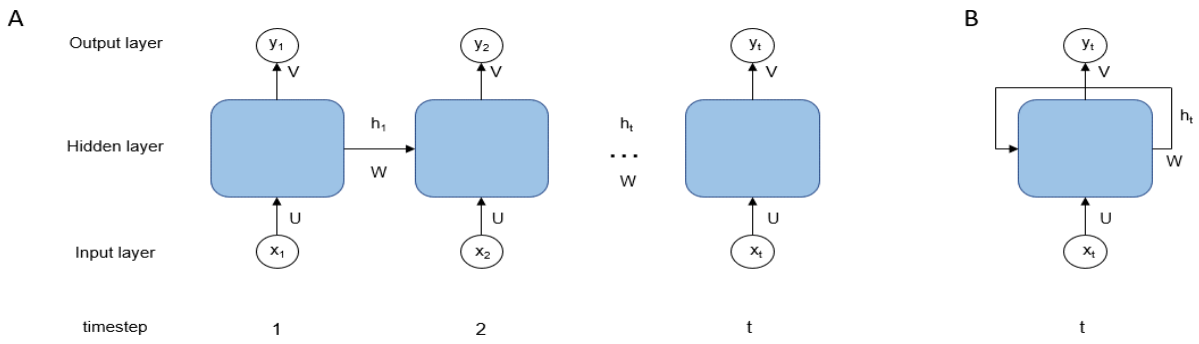


Figure 3-12. RNN Structure. **A,** Unrolled RNN structure. Where U, W, and V are weight parameters for the input, hidden, and output layers respectively. h_t is the cell state of the RNN which is updated at every timestep t. **B,** the rolled representation of an RNN's structure.

Despite the benefits of the original RNNs they only have a short-term memory, i.e. can only remember information close to the current timestep. This is due to the increasing difficulty of updating weights when using information that is further away from a current time step, known as the vanishing gradient problem. To overcome this, RNNs with gated cells were developed. This allows the control of information flow by selectively adding or removing information ultimately making training and capturing information much easier (by mitigating the issue of the vanishing gradient).

The most popular gated RNNs are the long short-term memory (LSTM) networks (Hochreiter and Schmidhuber, 1997). These contain cell states in addition to the hidden state. The cell state allows the gradient to flow uninterrupted during the backpropagation step of training.

The process of RNN-LSTM involves the following steps: 1) forgetting, 2) storing, 3) updating the internal cell state, and 4) generating an output. In step 1, intermediate inputs that do not carry significant semantic information are forgotten. In step 2, relevant information from the current input is stored. In step 3, the cell state is updated selectively. Finally, in step 4 the output gate returns a filtered version of the cell state.

3.7.3.1.2 Application in De novo Design

Distribution-learning

Several investigations into how the RNN workflow can be improved has been proposed. Gupta et al. (2018) identified that a higher sampling temperature in RNN-LSTM's decoding strategy increased the structural diversity but decreased the proportion of validity of the generated SMILES strings, and a lower sampling temperature reduced structural diversity. Li et al., (2020) suggested that performing virtual screening on RNN generated molecules improved novelty of the final molecules by exploiting the neighbouring chemical space of the generated molecules. The virtual screening was based on pharmacophore models and molecular docking.

Several studies have finetuned pretrained RNNs with a smaller dataset of active compounds (Yuan *et al.*, 2017; Gupta *et al.*, 2018; Merk *et al.*, 2018; Segler *et al.*, 2018; Tan *et al.*, 2020). For example, in Tan et al. (2020) the authors were able to generated novel compounds against schizophrenia, with validated polypharmacological effects on animal models, by training on known polypharmacological drugs.

Goal-directed generation

In terms of the optimisation used for the goal-directed generation of RNNs, several RL adaptations have been proposed. Early adaptations include the policy-based RL models such as REINVENT (Blaschke *et al.*, 2020; Loeffler *et al.*, 2024), ReLeaSE (Popova, Isayev and Tropsha, 2018) and DrugEX (Liu *et al.*, 2021; Šícho *et al.*, 2023); and the actor-critic RL model DeepFMPO (Ståhl *et al.*, 2019).

A simple implementation for goal-based RNN for *de novo* design is ReLeaSE, which first pretrains an RNN on a large SMILES dataset, i.e. ChEMBL, before using the same pretrained RNN for goal-based training using RL (Figure 3-13A) (Popova, Isayev and Tropsha, 2018). Results indicate optimisation was possible that were either biased towards a physical property (e.g. logP), bioactivity (e.g. JAK2), or structure complexity (e.g. number rings and substituents).

Later studies have investigated the use of two RNNs for goal-directed generation, which includes REINVENT and DrugEx. REINVENT is composed of two RNNs, termed the Prior and Agent networks, that are trained separately (Figure 3-13B) (Olivecrona *et al.*, 2017). The Prior is first pretrained on ChEMBL and subsequently used to initialise the Agent's training. During RL, the goal of the Agent is to generate molecules that move towards a user defined scoring function whilst being anchored by the learned distribution from the Prior. Here, only the Agent is finetuned whilst the Prior's learned weights and biases are frozen. These components provide the training loss, called augmented likelihood, that the Agent must optimise during finetuning. Results suggests that REINVENT is able to generate compounds that follow some user-defined scoring function. Investigated scoring functions include metrics that reward the agent when the molecule contains no sulphur S atom, when they contain an analogue of a query structure, and those that are predicted to be active against DRD2.

DrugEx use two distinct RNNs: the exploitation network G_θ and the exploration network G_ϕ (Liu *et al.*, 2019). The exploitation network is the primary generator trained with RL to optimise a given scoring function. In contrast, the exploration network is a fixed pretrained RNN that introduces variation in the token decoding process. This dual network approach operates collaboratively during training, where a random exploration rate ε determines which RNN to use for token selection. The aim here is to improve the diversity of the molecules produced. After training is completed, only the exploitation network remains to generate new molecules. This ensures that the final model is optimised for producing optimal candidates whilst having the benefits from the exploratory phase during training. The difference between REINVENT and DrugEx is REINVENT uses the RNNs to calculate the augmented likelihood loss used to finetune the model. On the otherhand, DrugEX uses a second RNN to modify the selection process during the token selection as a SMILES is being generated.

The Deep Fragment-based Multi-parameter Optimization (DeepFMPO) framework perform exploitation around the chemical space of a given query molecule (Ståhl *et al.*, 2019). The novel approaches that DeepFMPO introduces for generative *de novo* design is the use of bidirectional RNNs actor and critic, and the use of fragment-based construction (Figure 3-13C). Instead of individual symbols as is the case with SMILES, the inputs/outputs of the model rely on a library of fragments that are encoded into binary strings. Bidirectional networks are models that consists of two RNNs, where one RNN reads a sequence from start to end and the other reads it from the end to the start. This bidirectionality make it possible for an RNN to output a character based on both past information and subsequent steps. On the other hand, Ståhl et al. (2019) suggest that using fragment-based rather than atom-based construction prevents the model from generating molecules with long strings of carbons which can occur sincne the SMILES strings that are used for training typically contain many aliphatic carbons. The authors suggest that the model was able to design molecules that satisfy a set of criteria, which include molecular weight, clogP, and PSA.

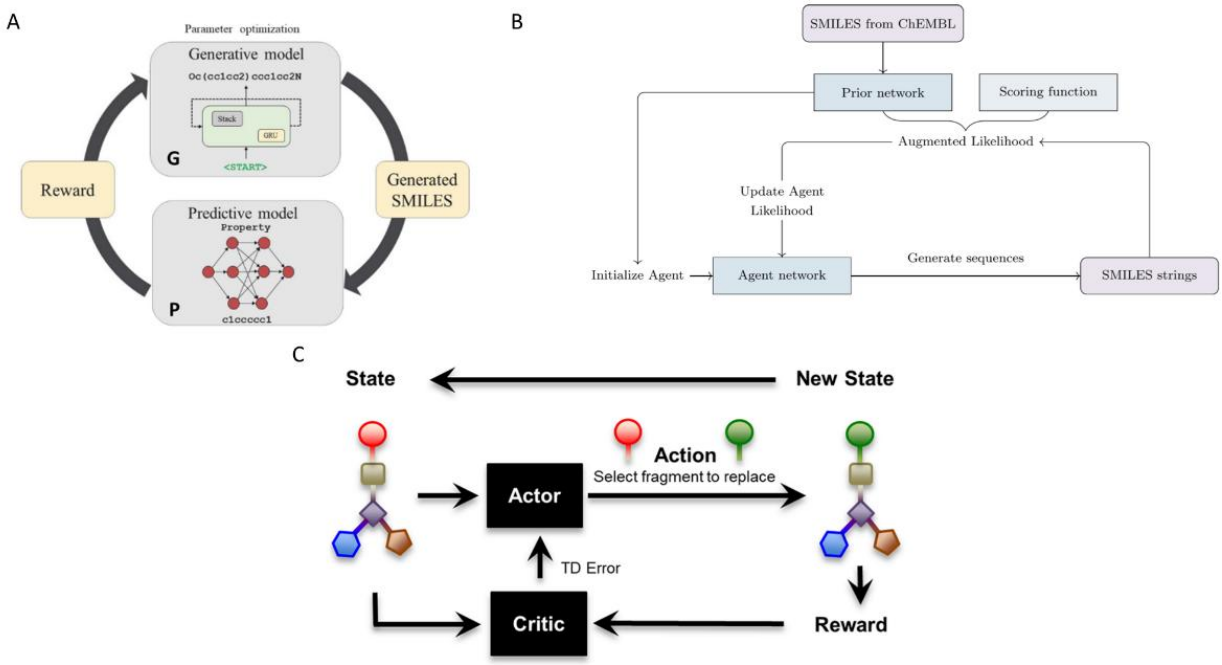


Figure 3-13. Reinforcement Learning-based RNNs. A, The ReLeaSE framework. Figure taken directly from (Popova et al., 2018). **B,** The REINVENT framework. Figure taken directly from Olivecrona et al. (2017). **C,** The DeepFMPO framework. Figure taken directly from Ståhl et al. (2019).

3.7.3.2 Transformers

Transformers were introduced by Vaswani et al. (2017) in the "Attention is All You Need" paper. This introduced a model that use self-attention instead of recurrence to handle input data. The main advantage of self-attention is that it can handle longer sequences much better. Despite the introduction of gated cells for RNNs, they are still limited by sequence length and are thus prone to the gradients vanishing or becoming explosively large. This leads to a loss in information from earlier time steps. Additionally, transformers allow the parallelisation of tokens, in contrast to RNNs which process sequences one token at a time. This allows transformers to be highly parallelisable.

There are different types of transformers, but they can generally be categorised into: Encoder-Decoder, Encoder-Only and Decoder-Only models. This section will focus on the Decoder-Only models only as they are the type mainly used for text generation and *de novo* design (Figure 3-14). A seminal example of this is the GPT transformer, which is the model behind ChatGPT (Radford *et al.*, 2019).

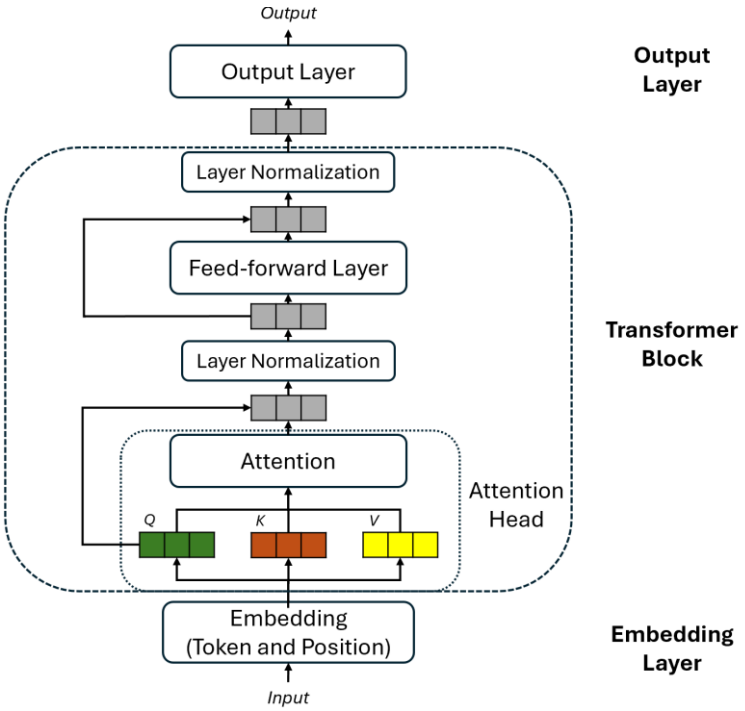


Figure 3-14. Decoder-only Transformer Architecture. The model's main component are an embedding layer, a transformer block, and an output layer.

3.7.3.2.1 Decoder-only Transformer Architecture

Decoder-only transformers have been specifically designed for generating sequences one token at a time. The key components of which include an embedding layer, transformer block, and an output layer. The important method here is the self-attention. This process checks the relationship between the current token and every single preceding tokens before it to make a prediction for the next token to generate. This is in contrast to RNNs, where the current token's relationship is compared with the hidden state, i.e. the compressed and combined summary of the preceding tokens, to make a prediction on the next token. The advantage of self-attention, therefore, is that information from a token that is far from the current token is not lost. This is particularly important when generated texts are long.

Embedding Layer

The input sequence is first converted into a fixed-dimensional space using an embedding layer. This converts discrete tokens into continuous vectors. Positional encodings are then added to the input embeddings to keep track of token order.

Transformer Block

A transformer block contains a multi-head self-attention layer and a position-wise feed-forward neural network. The final output of a transformer block is a contextualised vector representation. Multiple transformer blocks may be stacked on top of each other, where the output of one block is simply fed into the next block.

Attention Layer

The self-attention mechanism allows transformer models to weigh the importance of the different tokens within the input sequence. This means that the model can focus on relevant tokens when making predictions. In contrast to RNNs, instead of using a single hidden state, a hidden state is created for each token.

The process of self-attention involves computing a weighted sum of the hidden states of all the tokens in the sequence, where the weights are calculated from the similarity scores between tokens. To calculate a single attention head, the input embedding first goes through a learned linear transformation *W* to calculate the Query (Q) vector, which represent the current token; Key (K) vectors, which represent each token(s) in the sequence; and Value (V) vectors, which represent each token(s) in the sequence (Equations (5)-(7)).

$$Q = XW_Q \tag{5}$$

$$K = XW_K \tag{6}$$

$$V = XW_V \tag{7}$$

Attention scores are calculated using the dot products of the Query with all the Keys QK^T which are subsequently scaled by the square root of the Key dimension vector d_k (8). Attention weights are then calculated by applying a SoftMax function on the scaled attention score. To get the final output, the attention weights are multiplied by the V vector.

$$Scaled\ Attention\ Score = \frac{QK^T}{\sqrt{d_k}} \tag{8}$$

Multiple attention heads for a given input may be calculated in parallel, where heads have separate learnable parameters for producing the K, Q, and V vectors. The independent outputs of the different heads are then concatenated and are linearly transformed with W_O to project the vector into the desired output dimension, usually the same as the input dimension of the query d_q. Having multiple attention heads allows the model to encode different types of relationships between words. For example, one head might focus on syntactic relationships, while another focuses on semantic relationships

Feed-forward Network (FFN)

The output from the attention layer is then fed into a feed-forward neural network. This typically consists of two hidden layers with a ReLU activation function in between.

Layer Normalization and Residual Connections

The transformer block also uses layer normalization and skip connections. Layer normalization stabilises training by normalising each input to have zero mean and unit variance. Skip connections help limit the vanishing gradient problem during training by passing a tensor straight to the next layer and adding it to the processed tensor. There are many ways in which to arrange these components within a transformer block. The figure just shows one way.

Output Layer

The output vector from the transformer block is transformed into a probability of next token prediction. This process involves feeding the output vector into a linear layer, which transforms the output into logits over a token vector. Finally, a SoftMax function then converts these into a probability of next token predictions.

3.7.3.2.2 Application in De novo Design

One of the first transformers used for *de novo* design is by Adilov (2021). Their model is based on GPT that was pretrained on 5M SMILES strings from PubChem. They then added an adapter module, which consists of lightweight networks between separate transformer blocks. These allow the model to be trained on a task-specific situations without the need to finetune the entire transformer model. This is beneficial since it reduces computational cost. In the paper, the adapter models were finetuned on 5 active molecules from the TRPM8 dataset for distribution learning. The hypothesis is that the adapter models are lightweight, and the majority of the chemical knowledge is already encoded in the pretrained GPT model. Therefore, adapters should be efficient for fine-tuning even with a small dataset. The model was able to produce valid, novel, and diverse results. However, RNN-LSTMs have still been found to outperform transformers. Therefore, the RNN-LSTM is still the main model used in the popular generative *de novo* design model REINVENT (Nittinger, 2023).

3.8 Benchmarks

As the number of proposed approaches for generative *de novo* design is increasing, it is important to have benchmarks to assess and validate different generative models in a meaningful manner. Benchmarks may include suggestions for best practices in model development, hyperparameter tuning, etc. Additionally, benchmarks may provide standardised datasets, standardised models, and/or evaluation methods (distribution-learning or goal-directed) for comparison of different generative models. Having such benchmarks available will help comparison of the strengths and weaknesses of different generative *de novo* design models and may help indicate how they can be improved upon. These benefits have already been seen in other fields of machine learning, such as computer vision, where rapid progress has been made due to availability of standardised datasets such as MNIST (a database of handwritten digits) and CIFAR-10 (a database containing images that belong to 10 classes of objects, e.g. dog, bird, automobile, and etc).

The most popular benchmarking platforms for generative de novo design are: Molecular Sets (MOSES) (Polykovskiy *et al.*, 2020), GuacaMol (Brown *et al.*, 2019), Practical Molecular Optimization (PMO) (Gao *et al.*, 2022), and MolScore (Thomas *et al.*, 2024)

The benchmarks provide standardised datasets, baseline models, and a set of evaluation metrics but there are slight differences. In terms of evaluation metrics - MOSES provides an extensive set of distribution-learning benchmarks, GuacaMol provides both distribution-learning and an extensive set of goal-directed benchmarks, and PMO assesses models on 23 different tasks with a constraint of 10,000 scoring function (oracle) calls – the aim of which is to identify the most sample efficient models. The latter is particularly important for scoring function that are computationally demanding, such as docking. Additionally, MolScore provides benchmark tasks based on QSAR and docking. This is particularly important for evaluating generative models on more realistic tasks rather than relying on similarity-based such as the earlier benchmarks.

Limitations

Methods of evaluating generative methods have limitations, both in terms of distribution-learning and goal-directed benchmarks. In general, current metrics for both use cases are either not relevant for practical drug design or are trivial.

The main issue identified with distribution-learning benchmarks is that the evaluation metrics proposed from GuacaMol were easily fooled with a naïve AddCarbon model, where a model samples the training data and simply adds a carbon, except for the Fréchet ChemNet Distance metric (FCD) - a metric that compares the how similar the distribution of the training set is to the generated set of molecules using a neural network (Preuer *et al.*, 2018). Additionally, the AddCarbon model outperformed the baseline models included in GuacaMol, apart from RNNs. This highlights the problem in distribution-learning benchmarks whereby distribution-learning metrics can be maximised just with the random addition of carbon to the molecules in the training set (Renz *et al.*, 2019).

In terms of goal-directed benchmarks, the main issue identified was defining a suitable scoring function (Renz *et al.*, 2019). Using a scoring function based on ML has been shown to have data-specific biases, whereby models are optimised to produce similar molecules to the training set used to train the ML classifier. This suggest that current scoring methods limit the ability of models to explore the chemical space. As a result of the current limitations in computationally evaluating generative models, synthesis and biological evaluation remain the gold-standard (Walters and Murcko, 2020). This highlights the need for a more robust and efficient scoring function, which could additionally provide developments in polypharmacology modelling (Thomas *et al.*, 2022).

3.9 Conclusions

The first section of this chapter described the core components of a *de novo* design algorithm which includes assembly, scoring, and optimisation. The next section discussed generative deep learning and the fundamental elements involved. The subsequent section then contextualises generative deep learning into their application to *de novo* design. Finally, the benchmarks that have been developed to evaluate generative *de novo* design models were explored and their current limitations discussed. The next chapter on data augmentation describes a potential method that addresses some of the aforementioned limitations.

Chapter 4. Data Augmentation

4.1 Introduction

Many drug discovery tasks often face a lack of sufficient and/or quality data which limits the performance of machine learning models, such as out-of-domain generalisation (Tossou *et al.*, 2024; Chen *et al.*, 2025). Limited drug discovery data may be due to cost and time constraints. For example, public datasets may have missing experimental labels, i.e. they are sparse. This has been shown to negatively affect performance of multitask QSAR models (de la Vega de León et al., 2018).

The lack of quality data can be due to the information loss when representing molecules in computer-readable format, limited structural diversity, and missing negative data (Van Tilborg *et al.*, 2024). Additionally, the later stages of the drug discovery process, such as lead optimization, are inherently a low data problem and therefore cause issues in ML models (Altae-Tran et al., 2017). For example, the use of small datasets has been shown to cause overfitting of generative de novo design models, leading to low diversity in the generated set of molecules (Amabilino et al., 2020). The issues found in the low-data regime limits the use of generative de novo design models.

Several approaches have been proposed to address the issues of applying deep learning methods in low-data drug discovery scenarios (reviewed Van Tilborg et al., 2024). First, data augmentation, a popular method in the wider AI domain where the training data is artificially increased using the available dataset. An example of data augmentation in chemoinformatics is using randomised SMILES strings. Secondly, multi-stage training is a method where the model learns iteratively rather than in 'one-go'. For example, in active learning, a model is first trained on a small subset of the training dataset. A strategy is then used to select the most informative or diverse datapoints to be added to the training set over iterations. Finally, context-enrichment is where additional information is provided to the model which may include additional inputs. For example, multi-modal training takes advantage of using multiple molecular representations as input to improve model performance.

So far, alternative data augmentation techniques and active learning have not been explored extensively for *de novo* design. Developing data augmentation techniques for string-based molecular representations is difficult because syntax and semantics need to be considered. On the other hand, active learning requires both experimental and computational resources. Data augmentation is not explored widely in *de novo* design, especially goal-directed generation. Therefore, data augmentation provides an opportunity to develop novel methods.

Data augmentation is the process of artificially increasing the volume and/or diversity of data by modifying or creating synthetic versions of existing data. The aim of the method is to improve the generalisability and reliability of ML models, especially when training in the low data regime, by preventing overfitting. Additionally, it can be used to deal with class imbalance.

Multiple data augmentation methods have been proposed in NLP. These methods may provide alternatives to randomised SMILES for data augmentation in generative de novo design. In this chapter, we first describe the current literature for data augmentation in NLP and chemoinformatics. We then investigate how data augmentation techniques used in NLP may be applied to SMILES and how they impact generative de novo design model performance.

4.2 Data Augmentation in NLP

4.2.1 Paraphrasing-based Methods

In natural language, paraphrasing is a common way of expressing the same information using different words from the original text. Paraphrasing has therefore been used as a form of data augmentation by multiple studies. Examples of paraphrasing-based methods include thesauruses, semantic embeddings, and rule-based.

Thesauruses

Previous studies have investigated the replacement of words with their synonyms and hypernyms to obtain a semantically invariant transformation, i.e. generating a different expression whilst keeping the semantic meaning of the original text as much as possible.

A synonym is a word that has the same or similar meaning as another word. Synonyms are important in NLP as they allow algorithms to model the nuances of language more effectively. Studies using synonym replacement have used WordNet, a thesaurus that organises word synonyms according to similarity. A simple method of synonym replacement is to randomly select *n* words in a sentence that are not stop words and then replace them with random synonyms from WordNet (Wei and Zou, 2019; Zhang *et al.*, 2020).

Semantic Embeddings

Semantic embedding is a method using pre-trained word embeddings to generate semantically similar but syntactically different text from the original text data. Word embeddings are a numerical representation of words in a continuous vector of lower-dimensional space. These are learned representations from a large corpus of text using techniques like Word2Vec, Glove, and FastText. The aim of word embedding is to capture semantic relationships between words, so that words used in a similar context or that have similar meanings will have more similar vector representations. For example, the vectors for “cat” and “dog” will be more similar than “houses” and “skyscraper”. To use word embedding for data augmentation, original words in a sentence can be replaced by their k-nearest-neighbour words using cosine similarity (Wang and Yang, 2015).

Rule-based

Heuristics on natural language can be exploited to generate word-level and phrase-level paraphrases that retain sentence semantics. Regular expressions can be employed to alter the structure of text while preserving semantics, including modifications to verb forms, modal verbs, and negation. For example, “Admire” to “Admired” (Coulombe, 2018). Word pair dictionaries can also be used to substitute between the extended and abbreviated form of words (Regina, Meyer and Goutal, 2020).

Machine Translation

Machine translation is the method of translating from one language to another using computational techniques. It involves understanding of a source text in one language and generating an equivalent text in another language. Translation is therefore seen as a natural means of paraphrasing and has been used for data augmentation. A popular method of using machine translation for data augmentation is back-translation (Figure 4-1). Back-translation methods translate an original sentence from one language into another language and then back to the original language. As a result of back-translation, the original sentence is transformed into a similar, but not identical, sentence which introduces variation in the data. Back-translation does not directly replace individual words in a sentence but generates the whole sentence. Several studies have investigated back-translation between English and French as a form of data augmentation (Yu *et al.*, 2018; Fabbri *et al.*, 2020; Lowell *et al.*, 2020; Xie *et al.*, 2020; Zhang, Ge and Sun, 2020).

Figure 4-1. An Example of Back-Translation. Back-translation generates a semantically similar output to the input.

4.2.2 Noising-based Methods

Noise induction is the injection of random variations into existing data. In contrast to paraphrasing, where the focus is to keep the semantics of the original data as much as possible, noising-based methods perturb the original data to make the augmented set deviate from the original data. The advantage of this method is improving model robustness. The theory behind this method is that humans are capable of understanding text semantics even with slight noise such as spelling mistakes, slight changes in word order, etc. Slight noise should therefore improve model robustness (Li, Hou and Che, 2022).

Swapping

Swapping is exchanging the position of elements within text data. The method has been performed on multiple levels of data. For word-level swapping, the random swapping of word position in a sentence for *n* times ($n = \alpha l$ where α is a parameter that indicates the percentage of the sentence to be changed and l is the sentence length) has been investigated in multiple studies across a different range of tasks and models (Wei and Zou, 2019; Longpre, Wang and DuBois, 2020; Rastogi, Mofid and Hsiao, 2020; Zhang *et al.*, 2020).

Deletion

Deletion is the method of randomly removing each word or each sentence in text data given a probability *p*. This has been investigated for word-level (Wei and Zou, 2019; Longpre, Wang and DuBois, 2020; Peng *et al.*, 2020; Rastogi, Mofid and Hsiao, 2020; Zhang *et al.*, 2020), sentence-level (Yan *et al.*, 2019), and both word- and sentence-level (Yu *et al.*, 2019).

Insertion

Insertion is the method of inserting words or sentences in text data. For word-level insertion, a proposed method is where a random word that is not a stop in a sentence is selected, a random synonym of the selected word is then inserted in a random position within the sentence. This process is then repeated *n* times (Wei and Zou, 2019). For sentence-level insertion, a proposed method is the random selection of sentences from a different sample but with the same label (Yan *et al.*, 2019).

Substitution

Substitution is the method of randomly replacing words or sentences in text data. In contrast to the paraphrasing methods described above, substitution methods do not necessarily retain semantic similarity to the original text data. Different methods have been proposed to guide the substitution process. Original words or sentences in text data can be replaced with others within a task-related vocabulary (Xie *et al.*, 2017, 2020; Wang *et al.*, 2018; Daval-Frerot and Paris, 2020; Lowell *et al.*, 2020). Words can also be replaced with spelling mistakes. This may be based on a word bank of common spelling mistakes for, for example replacing “interesting” with “intresting” (Coulombe, 2018; Regina, Meyer and Goutal, 2020), or based on keyboard typos where a random letter in each word is replaced with an adjacent letter in a standard keyboard, e.g. “chemistry” with “chemistt” (Belinkov and Bisk, 2017). Finally, words can be randomly replaced a wildcard/placeholder such as “_” to mask information in the sentence (Xie *et al.*, 2017).

4.3 Data Augmentation in Chemoinformatics

Several studies have investigated the application of data augmentations of linear- and graph-based molecular representations in chemoinformatics, in the context of QSAR modelling and more recently generative de novo design (Bjerrum, 2017; Arús-Pous, Johansson, *et al.*, 2019; Magar *et al.*, 2021). These methods of data augmentation can be divided into data manipulation and alternative representation. Here we define alternative representation as methods that exploit the different ways a single unique molecule can be represented (Figure 4-2).

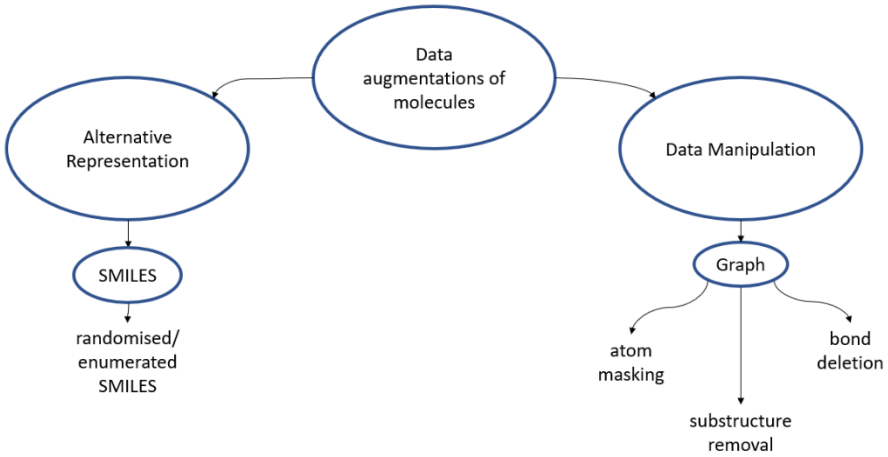


Figure 4-2. Types of Data Augmentations Suggested for Different Molecular Representations Used in Chemoinformatics.

QSAR Modelling

In QSAR, a simple approach that has been investigated is duplication of the input dataset. Results have shown improved model performance (Cortes-Ciriano and Bender, 2015; Kimber, Gagnebin and Volkamer, 2021). Later studies have developed more sophisticated data augmentation techniques for both linear- and graph-based notations.

For linear notation, randomised/enumerated SMILES has been proposed as a form of data augmentation which has been shown to improve model performance (Bjerrum, 2017). This type of data augmentation takes advantage of the different ways a molecule can be represented when performing graph traversal to produce SMILES strings (discussed in the Molecular Representations section) (Figure 4-3).

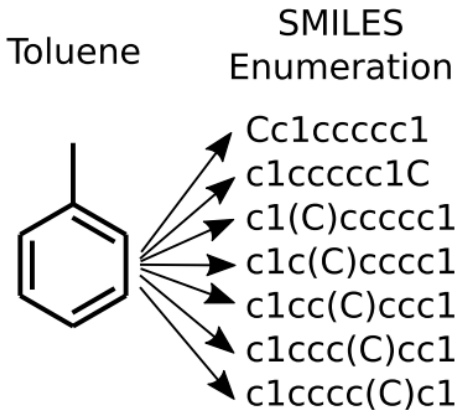


Figure 4-3. Different SMILES Representation of Toluene. The canonicalised SMILES string is the top string. Figure taken directly from Bjerrum (2017).

For graph-based representation, the open-source library AugLiChem was developed to provide a data augmentation framework for crystalline materials, fingerprints, and molecular graphs (Magar *et al.*, 2021). AugLiChem augments molecular graphs through atom masking, bond deletion, and substructure removal (Figure 4-4). In atom masking a node's feature is randomly replaced with a feature present in the database. In bond deletion, given a probability an edge is randomly removed from a molecule. The aim of these processes is to force the model to generalise what it has learnt from the molecular structures. For substructural removal, molecular graphs are created based on the decomposed fragments from BRICS. The aim of this is for the model to correlate target properties with functional groups. The generated molecules from augmentation are then given the same label as the original chemical instance. The authors suggests that the addition of augmented molecules into different GNN models for property prediction improved performance when compared to a baseline data.

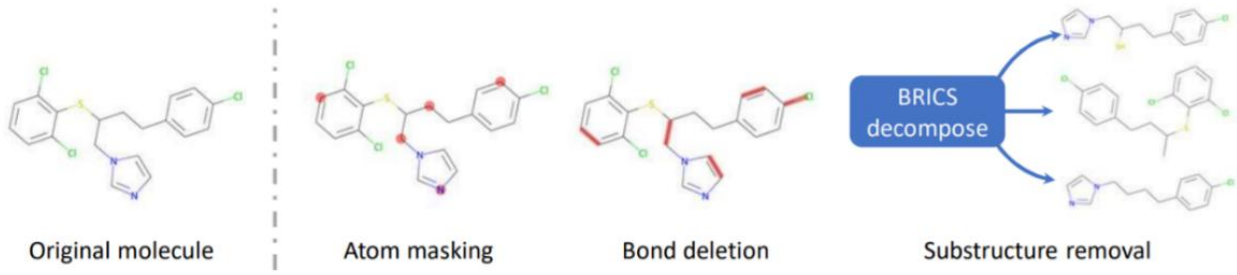


Figure 4-4. The AugLiChem Framework Augmentation Process for Molecular Graphs. Figure adapted from Magar et al. (2021).

De novo Design

It has shown that using randomised SMILES along with canonical SMILES to train an RNN for *de novo* design led to improved model performance (Arús-Pous, Blaschke, *et al.*, 2019). In the context of the study, data augmentation is when more than one type of representation of a single molecule is used for modelling training. Additionally, it was found that models trained on randomised SMILES performed better than those trained solely on canonical SMILES, where the latter led to premature overfitting. The authors suggest that it is more difficult for the model to learn the SMILES syntax when it must both learn how to generate valid SMILES strings and how to generate their canonicalized form. In comparison, randomised SMILES models are not limited by the canonical restriction. Interestingly, these findings have not been adopted widely by more recent studies as they tend to use canonicalised SMILES. Later studies have also improved performance when using randomised SMILES to augment RNNs in a goal-directed generation task (Bjerrum *et al.*, 2023; Guo and Schwaller, 2024).

A recent study has investigated noising-based methods to introduce synthetic data rather than providing an alternative representation of the same molecule using randomised SMILES. Brinkmann et al. (2025) introduced atom deletion, token deletion, and bioisosteric substitution for generative *de novo* design models tasked on distribution-learning. Other noising-based methods and their application to goal-directed generation remains novel and is the subject of investigation of this thesis.

4.4 Conclusion

This chapter explored data augmentation operators found in the NLP domain, which includes paraphrasing- and noising-based methods. The next section discussed the data augmentation methods that have been reported in chemoinformatics, both for QSAR and *de novo* design. This discussion has identified the lack of data augmentation strategies in this domain, especially for generative *de novo* design models tasked on goal-directed generation. The next chapter describes the methods that will form a core part of the investigations in the application of data augmentation in generative *de novo* design.

Chapter 5. Methods

5.1 Introduction

This chapter describes in detail the main methods that are used throughout the experimental chapters. The main methods include the GuacaMol benchmark, which is an open-source platform for generative *de novo* design that has been previously described in the literature review chapter. Additionally, the randomised SMILES implementation is described here as it provides a comparison for the methods developed in later chapters. This is the most widely used form of data augmentation in generative *de novo* design and goal-directed generation tasks, as described in the literature review chapters.

5.2 Python Packages

The methods described in this chapter were developed using Python 3.7.9. RDKit 2020.09.1.0 was used for handling of molecular data. NumPy 1.21.4 and Pandas 1.2.1. were used for general data processing.

5.3 GuacaMol Benchmark

The GuacaMol benchmark is an open-source platform for evaluating the performance of deep *de novo* design models on both distribution-learning and goal-directed generation tasks (available on: https://github.com/BenevolentAI/guacamol) (Brown *et al.*, 2019). It consists of a set of evaluation tasks, a standardised dataset, and various *de novo* design baseline models. These are discussed further below

The GuacaMol benchmark is based on Python 3.7 and uses several Python libraries to build and train *de novo* design models. RDKit 2020.09.1.0 was used for handling of molecular data. NumPy 1.21.4 and Pandas 1.2.1. were used for general data processing. Several Python libraries were used to extract data, handle the data, and perform visualisations. NumPy 1.22.3 and Pandas 1.2.5 were used for general data processing. Seaborn 0.10.1 was used for data visualisation.

5.3.1 Dataset

The dataset used for this study was obtained from the GuacaMol benchmark (Brown *et al.*, 2019). The benchmark dataset itself is derived from the ChEMBL24 dataset, which contains molecules that have been synthesised and tested against biological targets (*CHEMBL24*, 2018). In brief, how the training data was generated by Brown et al. (2019) is as follows: ChEMBL24 was pre-processed by removing salts, charges were neutralised, SMILES strings with more than 100 characters were removed , and molecules with atoms outside of H, B, C, N, O, F, Si, P, S, Cl, Se, Br, and I were removed. Additionally, molecules with an ECFP4 similarity greater than 0.323 compared to a holdout set consisting of the drugs used in the goal-directed benchmarks (celecoxib, aripiprazole, cobimetinib, osimertinib, troglitazone, ranolazine, thiothixene, albuterol, fexofenadine, and mestranol) were removed. The

remaining SMILES were canonicalised using RDKit. The resulting training dataset consists of 1.2M canonical SMILES molecules.

5.3.2 Goal-Directed Generation Tasks

The goal-driven tasks from the GuacaMol benchmark were used to evaluate the effect of data augmentation on generative *de novo* drug design performance (Brown *et al.*, 2019). GuacaMol's goal-driven tasks use multiple metrics to define how the generated molecules should be optimised. Generated molecules are scored on a combination of molecular features and/or other metrics, all of which have been implemented using RDKit. These metrics can be combined either through a geometric or arithmetic mean. Molecular features used in the benchmark include topological surface area (TPSA), a metric for predicting transport properties of drugs; partition coefficient (logP), a metric for measuring drug permeability in target tissues; Bertz, a metric that quantifies molecule complexity; and quantitative estimation of drug-likeness (QED), a metric that measures the underlying property distribution of a molecule. Other metrics that have been implemented include similarity and SMARTS scores. The similarity scoring function *sim(x, y)* evaluates the Tanimoto similarity of a molecule's fingerprint (ECFC6, ECFC4, or AP) *y* to the fingerprint of the target molecule *x*. ECFC (Extended Connectivity Fragment Count) is a circular fingerprint that counts the number of fragments around an atom up to a bond radius, here a radius of 6 (ECFC6) or 4 (ECFC4) were used. AP(Atom-Pair) is a topological fingerprint that encodes information about pairs of non-hydrogen atoms and the shortest bond distance between them within a molecule. The SMARTS scoring function *SMARTS(s, b)* evaluates whether a SMARTS pattern *s* is present, *b* = true, or not present, *b = false,* in a molecule. The isomer scoring function (x) evaluates the geometric mean of the contributions of the present element types and the total number of atoms in the molecule. The mentioned metrics can also be modified by various functions to provide flexibility on how the final molecule score is computed. Table 5-1 shows the goal-directed benchmark tasks available in GuacaMol and the scoring-functions used to construct them. A more detailed description is available from (Brown *et al.*, 2019).

Table 5-1. GuacaMol's Goal-Directed Benchmark Tasks. The scoring function indicates the properties that need to be optimised for a given task. Table adapted from Brown et al. (2019). The number of fluorine atoms tasks model to generate molecules that have exactly one F atom. The number of aromatic rings tasks models to generate molecules with exactly 2 aromatic rings. The number of rings tasks models to generate molecules with exactly 3 rings.

Benchmark Tasks	Scoring Function(s)
Celecoxib rediscovery	sim(celecoxib, ECFC4)
Troglitazone rediscovery	sim(troglitazone, ECFC4)
Thiothixene rediscovery	sim(thiothixene, ECFC4)
Aripiprazole similarity	sim(aripiprazole, ECFC4)
Albuterol similarity	sim(albuterol, FCFC4)
Mestranol similarity	sim(mestranol, AP)
C11H24	isomer(C11H24)
C9H10N2O2PF2Cl	isomer(C9H10N2O2PF2Cl)
Median molecules 1	sim(camphor, ECFC4), sim(menthol, ECFC4)
Median molecules 2	sim(tadalafil, ECFC6), sim(sildenafil, ECFC6)
Osimertinib MPO	sim(osimertinib, FCFC4), sim(osimertinib, ECFC6), TPSA, logP
Fexofenadine MPO	sim(fexofenadine, AP), TPSA, logP
Ranolazine MPO	sim(ranolazine, AP), logP, TPSA, number of fluorine atoms
Perindopril MPO	sim(perindopril, ECFC4), number aromatic rings
Amlodipine MPO	sim(amlodipine, ECFC4), number rings
Sitagliptin MPO	sim(sitagliptin, ECFC4), logP, TPSA, isomer(C16H15F6N5O)
Zaleplon MPO	sim(zaleplon, ECFC4), isomer(C19H17N3O2)
Valsartan SMARTS	SMARTS(predefined SMARTS 2, true), logP, TPSA, Bertz
deco hop	SMARTS(predefined SMARTS 2, true), SMARTS(predefined SMARTS 3, false), SMARTS(predefined SMARTS 4, false), sim(predefined SMILES 5, PHCO)
scaffold hop	SMARTS(predefined SMARTS 2, false), SMARTS(predefined SMARTS 6, true), sim(predefined SMILES 5, PHCO)

5.3.3 Recurrent Neural Network with Long Short-term Memory

GuacaMol includes a recurrent neural network with long short-term memory (RNN-LSTM) generative model that handles sequential data and is used to predict the next SMILES token given the current SMILES token and previous SMILES token. The application of the RNN-LSTM to *de novo* design follows a two-step training process, pretraining and finetuning. The aim of the pretraining process is to learn the syntax and semantics of SMILES representations of chemical structures. The subsequent step of finetuning then directs the learned rules into a chemical space with desired properties. Here, the aim is to generate molecules that progressively score higher using a user-defined score.

Primary Training Procedure – Distribution Learning (pretraining)

The GuacaMol benchmark provides an RNN-LSTM model which has been pretrained on the pre-processed ChEMBL24 dataset. The default model parameters set in GuacaMol is used throughout the experimental work described in the thesis: layers = 3, hidden layer size = 1024, batch size = 512, epochs = 10, dropout = 0.2, and a learning rate = $1x10^{-3}$.

Secondary Training Procedure – Goal-directed Generation (finetuning)

The pretrained RNN-LSTM can then be tasked to perform goal-directed generation by finetuning. The default GuacaMol finetuning parameters are used throughout: n-iterations = 20, m-sequences = 1024, and top-k = 512.

The finetuning technique used in GuacaMol is hill-climbing, also known as iterative fine-tuning. In this type of optimisation, the model is continually retrained on the top-k SMILES that score the highest on some user-defined scoring function at each epoch, i.e. loop. More formally, for n-iterations, the pretrained model generates m-sequences of SMILES which are subsequently scored using a user-defined scoring function and stored in an archive. The top-k SMILES from the archive are then used to retrain the model at each iteration (Figure 5-1). The pseudocode for the iterative fine-tuning is also described below (Neil *et al.*, 2018).

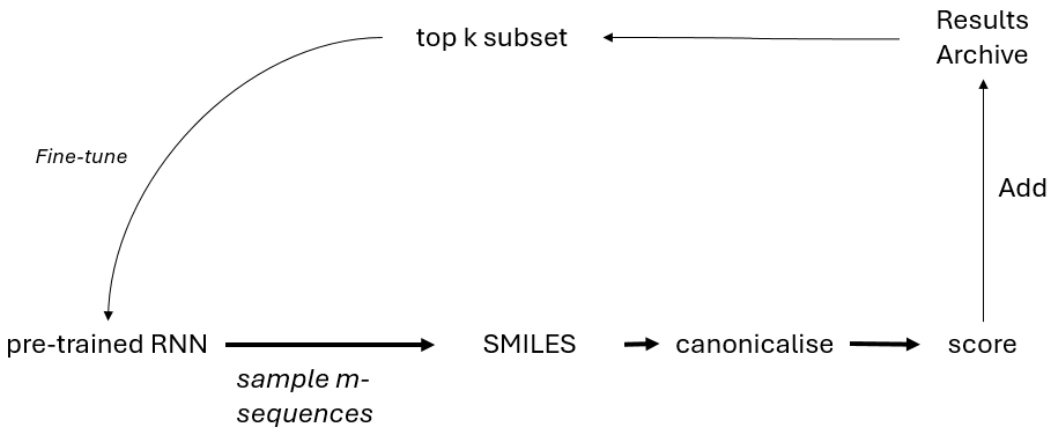


Figure 5-1. The Iterative Fine-Tuning Workflow. For n-iterations, a given pretrained model is sampled for m-sequences. The output SMILES are then scored and stored in an archive. The top-k SMILES from the results archive for a given iteration are then used to retrain the model.

```
1. For n-iterations do
2.     for m-sequences do
3.         Sample SMILES from model
4.         Calculate scores of the SMILES using a scoring function
5.     end for
6.     Store generated SMILES in an archive
7.     Take the top-k SMILES with the highest score from the archive
8.     Fine-tune the model on the top-k SMILES structure for n-epochs
9. end for
```

In the pseudocode, the n-iterations parameter determines the number of fine-tuning loops for a given model. The m-sequences parameter dictates how many SMILES are sampled, i.e. generated, from the model at a given iteration. Finally, the top-k SMILES dictates the number of top scoring SMILES used to finetune the model. Refer to Neil et al. (2018) for more details of the pseudocode and the original implementation.

A pretrained model may be sampled randomly for m-sequences, as it is, where the model will generate SMILES that follow the learned chemical space distribution during the pretraining step. This is called random start. However, it is likely that the molecules generated will have low scores for a scoring function of interest, especially for more complex objective functions. As a result, the model will require more iterations to move into a more desirable chemical space. This can be computationally expensive and result in longer time for convergence for an optimal solution. Instead, GuacaMol by default initialises the pretrained model prior to being sampled for goal-directed generation tasks.

Initialisation is performed by first scoring the GuacaMol training dataset with the benchmark task of interest. The top-1024 scoring molecules are then used to finetune the pretrained model. This then biases the model to initially generate molecules that follow the chemical distribution of the top-1024 molecules. The initialisation step will not lead to trivialising the benchmark tasks as molecules that have an ECFP4 similarity higher than 0.323 to the compounds used to define the similarity scoring functions in Table 5-1 have been removed. This prevents the model from generating molecules that are trivially similar to the compounds of interest.

5.3.4 Graph GA

Graph GA is a genetic algorithm that simulates evolution at the graph level. The implementation used by GuacaMol is from Jensen (2019). For a given goal-directed benchmark task, the algorithm selects the top 100 scoring molecules from the pre-processed ChEMBL24 dataset as the initial population.

At each generation, or epoch, a mating pool of 200 molecules is created by sampling from the population with replacement, where the probability of selecting a molecule is proportional to its score. Sampling with replacement means that the same molecule can be drawn multiple times: high-scoring molecules are therefore more likely to appear multiple times in the mating pool, while low-scoring molecules may appear only rarely or not at all. A new population of 100 molecules is then generated by randomly selecting pairs of molecules from the mating pool and applying a crossover operation, with a chance of performing a mutation operation. This process continues for 1000 iterations or until there is no progress for five consecutive epochs.

The crossover operation is based on fragmenting and recombining two parent molecules. The type of crossover performed is selected at random, with equal chance of selection. The first type is a non-ring crossover where fragments connected by acyclic bonds are exchanged. Alternatively, a ring crossover can be performed where entire or partial ring systems are exchanged by cleaving two bonds simultaneously. Additionally, there is a 50% chance that a mutation will be applied to each offspring molecule. The type of mutation operation is selected at random with equal probability of selection. The mutation operations include the append atom, insert atom, delete atom, change atom type, change bond order, delete ring bond, and add ring bond. RDKit's SMARTS patterns and SMARTS reactions are used to perform both the crossover and mutation operations above.

5.4 Randomised SMILES

As discussed in the literature review chapter, SMILES randomisation, or enumeration, is the first form of data augmentation that has been explored in *de novo* drug design (Arús-Pous, Johansson, *et al.*, 2019; Bjerrum *et al.*, 2023; Guo and Schwaller, 2024). These prior works have shown that SMILES randomisation helped improve model performance of *de novo* design models, and therefore, SMILES randomisation was used as a comparison for the data augmentation methods explored in the thesis.

The algorithm from Arús-Pous et al. (2019) available at https://github.com/undeadpixel/reinvent-randomized was used to generated randomised SMILES. Briefly, randomised SMILES can be generated by completely randomising the atom numbering or by randomising the atom numbering with restriction within a molecule. Randomising with restriction involves using RDKit to prevent atom numbering that may produce 'strange' or overly complicated SMILES strings, for example, by prioritising traversing sidechains. In both cases, how a molecule is traversed during SMILES creation is changed and therefore different SMILES representations are created for a given molecule. SMILES randomisation with restriction is used as it was shown to improve generative *de novo* design performance in contrast to the unrestricted randomisation (Arús-Pous, Johansson, *et al.*, 2019).

5.5 Statistical Significance

The pairwise Welch's t-test from SciPy was used to determine if the differences in the scores (MPO, SAscore, and QED) of the molecules generated between the generative *de novo* design models were statistically significant. This statistical measure was used because there are unequal variance and sample sizes between the generated molecules of the different models. Welch's t-test is more robust in this scenario compared to the student's t-test and ANOVA (Delacre, Lakens and Leys, 2017).

Initially, for each pair of models, the null hypothesis is assumed which means there are no significant differences between the scores of the generated molecules. The alternative hypothesis is that a significant difference does exist. The threshold for statistical significance was set at 0.05. If a test resulted in a p value less than the threshold, then the null hypothesis is rejected and the alternative hypothesis is accepted. A smaller p-value indicate stronger evidence against the null hypothesis; this is denoted by statistical significance. The levels of statistical significance were also categorised as follows: **** ($p <= 0.0001$), *** ($p <= 0.001$), ** ($p <= 0.01$), * ($p <= 0.05$), and ns for not significant ($p > 0.05$).

Chapter 6. Primary Dataset Augmentation

6.1 Introduction

A recent trend in *de novo* design is using deep learning models. These models have the advantage of learning complex and highly non-linear patterns (Lecun, Bengio and Hinton, 2015). However, deep learning algorithms require large datasets to perform well. This presents a problem when applying deep learning in drug discovery as it is a field that typically deals with low data because of the small number of known actives available (Waring *et al.*, 2015). Training on a small dataset has been found to negatively impact deep *de novo* design models as it leads to overfitting which causes reduced diversity of the generated molecules (Amabilino *et al.*, 2020) and reduced validity and chemical space coverage (Arús-Pous, Johansson, *et al.*, 2019). A potential solution for this is to use data augmentation (Van Tilborg *et al.*, 2024).

The application of data augmentation in *de novo* design remains limited. As discussed in the literature review chapter, currently, data augmentation has been applied to the primary training dataset to improve distribution learning performance and the secondary training dataset to improve goal-directed generation performance. However, it is unknown how the data augmentation of the primary dataset affects model performance on goal-directed generation. This is an important consideration as the primary training dataset is important to learning a broad chemical knowledge. Data augmentation may provide a more diverse primary training dataset and therefore the pretrained model can learn a richer “vocabulary”, i.e. scaffolds and building blocks.

Generative de novo design typically undergoes a two-stage training process, pretraining and finetuning. Pretraining involves training generative models on a large, general-purpose dataset, termed here as the primary training dataset. The goal of this first training step is to learn the grammar and syntax of the chemical language, such as SMILES string representation, without being biased towards any specific property or biological target.

Finetuning involves training the pretrained generative models on a smaller, curated dataset, termed here as the secondary training dataset. The aim at this step is to adapt the learned chemical knowledge during pretraining to generate molecules with specific properties. The finetuning step can be further subdivided into static and dynamic training dataset depending on the task. A static secondary training dataset is used for distribution learning. Here, the pretrained model is further trained to learn the distribution of properties of a human-curated dataset. For example, collecting known actives against a specific drug target (Amabilino *et al.*, 2020). The aim here is to learn the chemical patterns that are characteristic of the known actives. On the other hand, a dynamic secondary training dataset is used for goal-directed generation (Neil *et al.*, 2018; Blaschke *et al.*, 2020). Goal-directed generation algorithms undergo optimisation loops where high scoring molecules generated at each loop are used to finetune the model towards the ideal chemical space. Therefore, the set of molecules are used to further train the model may be different at each optimisation loop. This is in contrast to the static dataset, where the dataset is predefined and does not change.

The first section of this chapter describes the data augmentation that has been reported in the literature but has not been applied in generative *de novo* design and/or augmenting the primary training dataset to improve goal-directed generation performance. The next section investigates the effect of the above data augmentation methods on goal-directed generation when applied to the primary training dataset.

6.2 Methods

6.2.1 Data Augmentation

Data augmentation is categorised here into methods that produce synthetic data that are naïve, different representation, or producing an entirely new sample. Naïve data augmentation are methods that do not add new information or diversity to the dataset. Data augmentation that produces different representations of the same sample aim to transform a sample's features without changing its label and meaning. Finally, data augmentation that produces new samples aims to introduce a wider range of data variations to a model.

Naïve - Duplication

Duplication is a naive method where SMILES from the original dataset are copied exactly to increase the dataset size with *N* repetitions/fold. The potential application of duplication for data augmentation in generative *de novo* design has not been previously investigated and is therefore explored in this chapter.

In this work, an augmented dataset that was created by applying duplication with 1 repetition on the GuacaMol dataset (1.2 million SMILES) contains the original dataset and a complete copy of the original data. Therefore, the augmented dataset contains 2.4 million SMILES (GuacaMol dataset + GuacaMol dataset) such that each SMILES in the original dataset is present twice in the augmented dataset. The present work performs 1 repetition of the original GuacaMol dataset (a x1-fold augmentation) as it was previously identified that more repetition caused a decreased in QSAR performance (Cortes-Ciriano and Bender, 2015).

Different Representation - Randomised SMILES

Randomised, or enumerated, SMILES takes advantage of the different ways the same molecule can be represented during SMILES generation. Randomised SMILES have been shown to improve distribution learning performance when used to augment the primary training dataset (Arús-Pous et al., 2019) and goal-directed generation performance when augmenting the secondary training dataset (Bjerrum *et al.*, 2023; Guo and Schwaller, 2024). However, the effect of augmenting the primary training dataset on goal-directed generation performance has not been explored previously.

The algorithm from Arús-Pous et al. (2019), available at https://github.com/undeadpixel/reinvent-randomized, was used to generate randomised SMILES. Briefly, randomised SMILES can be generated by completely randomising the atom numbering or by randomising the atom numbering with restriction within a molecule. Randomising with restriction involves using RDKit to prevent atom numbering that may produce 'strange' or overly complicated SMILES strings, for example, by prioritising traversing sidechains. In both cases, how a molecule is traversed during SMILES creation is changed and therefore different SMILES representations are created for a given molecule.

The present work uses randomisation with restriction as it was shown to improve model performance the best in generative *de novo* design tasked on distribution learning (Arús-Pous et al., 2019). Additionally, a x1-fold augmentation was used to create an augmented dataset which contains the canonicalized SMILES strings plus one alternative representation.

New Sample - BRICS Enumeration

In NLP, a common form of data augmentation is the random swapping of word position in a sentence (Wei and Zou, 2019; Li, Hou and Che, 2022). A recent study adapted this concept to ML-based polymer property prediction where it is called iterative rearrangement (Lo *et al.*, 2023). Inspired by these data augmentation methods, BRICS enumeration is proposed here as a form of data augmentation for generative *de novo* design (Figure 6-1). This may also have applications for QSAR but is beyond the scope of the thesis.

The first step of the algorithm involves RDKit's BRICSDecompose() function. The function is based on an implementation of BRICS (breaking of retrosynthetically interesting chemical substructures) which is a popular method for automated retrosynthetic fragmentation to extract chemical motifs from molecules of interest (Degen *et al.*, 2008). The next step involves RDKit's BRICSBuild() function which enumerates all possible fragment recombinations. Experiments involving BRICS enumeration were restrained due to computational constraints. First, only 0.2%, 1%, and 10% of the original primary data was chosen at random for BRICS enumeration. Second, a maximum of 5 enumeration samples were obtained for each molecule. Note, that since some molecules may contain only a few BRICS fragments, e.g. small molecules, some augmentation may produce fewer than 5 enumeration samples.

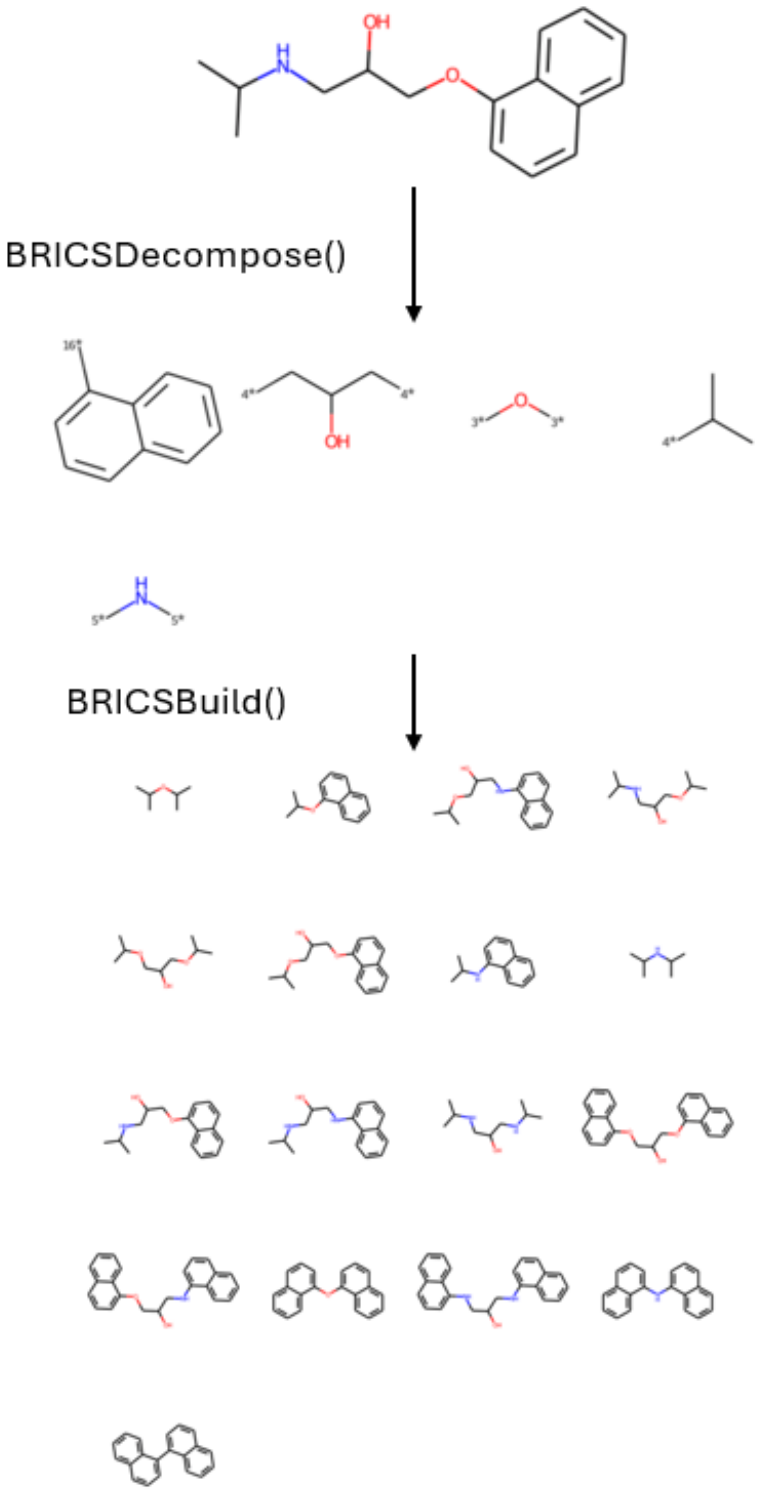


Figure 6-1. BRICS Enumeration Method for Data Augmentation. The first step involves using RDKit's BRICSDecompose. The resulting BRICS fragments can then be recombined using RDKit's BRICSBuild() function to provide new molecules which represent enumerated combinations of the fragments.

6.2.2 GuacaMol Benchmark

The GuacaMol benchmark was used to evaluate the performance of the generative models (available on: https://github.com/BenevolentAI/guacamol) (Brown *et al.*, 2019). GuacaMol also provides a standardised dataset and various *de novo* design models.

6.2.2.1 Dataset

The dataset used for this study was obtained from the GuacaMol benchmark (Brown *et al.*, 2019). The benchmark dataset itself is derived from the ChEMBL24 dataset, which contains molecules that have been synthesised and tested against biological targets (*CHEMBL24*, 2018).

6.2.2.2 SMILES RNN-LSTM

The GuacaMol benchmark provides an RNN-LSTM model which was pretrained on the pre-processed ChEMBL24 dataset. The default model parameters used in GuacaMol are layers = 3, hidden layer size = 1024, batch size = 512, epochs = 10, dropout = 0.2, and a learning rate = $1x10^{-3}$.

6.2.2.3 Gaol-directed Generation Tasks

In this chapter, the full set of GuacaMol benchmark's goal-driven tasks were used. This includes: rediscovery and similarity to a hold-out drug molecule; median molecules, where the aim of the model is to generate molecules that are similar to two target molecules simultaneously; MPO benchmarks, where the aim of the model is to generate molecules that fit multiple molecular features and other metrics; SMARTS benchmark, where the aim of the model is to generate molecules that contain valsartan SMARTS and have the properties similar to sitagliptin; and scaffold/decorator hop benchmarks, where the aim of the model is to maximise similarity or dissimilarity to a scaffold whilst keeping specific substituents.

6.2.2.4 GuacaMol Task Score Trajectory vs Epoch Analysis

The efficiency of de novo design models in finding optimal solutions is an important consideration, termed sample efficiency. A model that generates high scoring molecules at an earlier epoch means fewer training epochs and therefore require fewer generation-scoring cycle. A sample efficient model is therefore more cost-effective (Gao *et al.*, 2022).

The score trajectories were plotted as a function of the number of epochs. At each epoch, the score of the top-1 molecule was obtained. This was then averaged across repeats. Additionally, the training and validation losses were plotted as a function of the number of epochs to identify whether data augmentation makes training easier or more difficult. Training loss indicates model performance on the training data and is used to update model parameters. On the other hand, validation loss indicates how well the model generalises to unseen data. This is used to monitor potential model overfitting. Like the top-1 scores, these were averaged across repeats.

A “solvable” task was chosen for the trajectory analysis because it shifts the evaluation from whether a model can generate ideal molecules into how efficiently it generates ideal molecules. Celecoxib rediscovery was chosen as an example as it is one of the task where the model fully optimises the score. Additionally, in rediscovery tasks models must aim to generate a specific molecule. Therefore, there is a known end point, i.e. generating a specific molecule.

6.2.3 Experimental Workflow

Figure 6-2 shows the experimental workflow for this chapter. The RNN-LSTM was pretrained on different datasets to assess how augmenting the pretraining dataset affects goal-directed generation performance:

- The baseline dataset (1.2M SMILES) is the original training dataset provided by GuacaMol.

- The randomised dataset (1.2M SMILES) replaces each SMILES from the baseline dataset with a randomly created SMILES representation. The +randomised dataset (2.4M SMILES) augments the baseline dataset by generating one randomised SMILES representation for each of the baseline SMILES.
- The +baseline dataset is the duplicated dataset (2.4M SMILES), where each of the baseline SMILES has been copied and added to the dataset.
- The +BRICS dataset is the GuacaMol dataset with the addition of molecules generated from BRICS enumeration. The starting population for the BRICS enumeration are a subset of the GuacaMol dataset and includes the sizes of 3k, 12k, and 120k.

Finally, the RNN-LSTM models that were pretrained on different datasets were evaluated on the full suite of GuacaMol's goal-directed generation tasks.

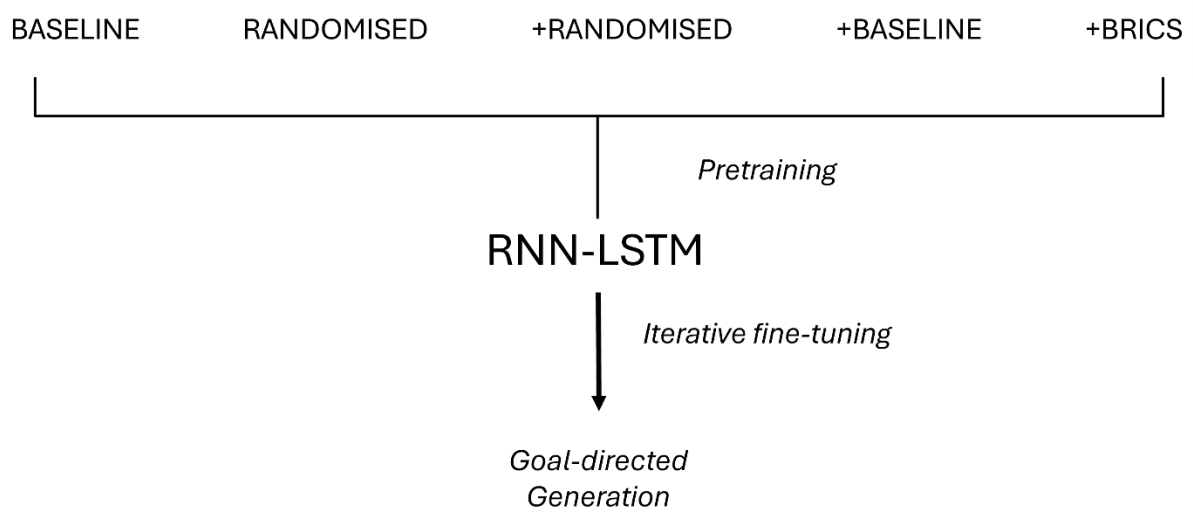


Figure 6-2. Experimental Workflow of the Data Augmentation of the Primary Dataset Experiments. The RNN-LSTM model was pretrained on different datasets. The models were then evaluated on GuacaMol's goal-directed generation tasks.

6.3 Results and Discussion

6.3.1 Performance on GuacaMol's Goal-directed Generation Task

Duplication and Randomised Data Augmentation

Table ***6-1*** shows the goal-directed generation performance of the RNN-LSTM pretrained on different datasets. Baseline indicates the model being pretrained on the original GuacaMol training dataset (1.2M). RNN1 indicates the model being trained on the randomised SMILES representation rather than the canonical SMILES representation (1.2M). RNN2 indicates the model being trained on an augmented dataset, where a single randomised SMILES representation has been generated for each of the baseline samples and included in the overall dataset (2.4M). Finally, RNN3 indicates the model being trained on an augmented dataset, where each of the SMILES from the baseline has been copied once and included in the overall dataset (2.4M). Five RNN-LSTM models were trained on each of these datasets and their goal-directed generation performances were averaged across the repeats.

Results indicate that both duplication and randomised SMILES improved overall model performance for the goal-directed generation benchmark, with RNN1 performing the best (total mean = 17.193) compared to the baseline (total mean = 17.002). A potential hypothesis is that the model trained on canonicalised SMILES only needs to learn how to generate both valid and the canonical representation of a SMILES molecule (Arús-Pous, Johansson, *et al.*, 2019).

An interesting result is the RNN2, which is the canonicalised SMILES with an additional randomised SMILES for each molecule (i.e. a x1-fold data augmentation). The results show an increased model performance compared to the baseline. However, this contradicts a previous finding where it was found that a x1-fold augmentation decreased model performance (Moret *et al.*, 2020). The authors hypothesised that a single randomised SMILES representation is insufficient to accurately learn the SMILES grammar. A potential explanation for this discrepancy is that Moret et al. (2020) evaluated model performance based on % validity, % uniqueness, and % novelty (i.e. distribution learning benchmarks), whereas this work evaluated model performance based on goal-directed generation benchmarks. This may suggest that using data augmentation on the primary dataset has different effects on distribution learning benchmarks and goal-directed generation benchmarks.

Finally, results show that Sitagliptin MPO is a task where data augmentation did not help improve model performance. This suggests that this task is difficult for deep de novo design models. Finally, results indicate that x1 duplication of the original dataset did not make overall performance worse. This is in contrast with the findings of a previous work on using duplication as a form of data augmentation for QSAR, where it was found that duplications decreased model performance (Cortes-Ciriano and Bender, 2015).

Table 6-1. Goal-Directed Generation Scores for Duplication and Randomised SMILES Data Augmentation. Bold – greater than baseline, Red – less than baseline. Means are calculated from 5 repeats. GuacaMol training dataset size = 1,273,104. Models: Baseline (1.2M); RNN1: Replace all canonicalised SMILES with randomised SMILES (1.2M); RNN2: Baseline dataset + randomised SMILES from RNN1 (total 2.4M), RNN3: Baseline dataset + Baseline dataset (total 2.4M).

		Baseline	RNN1	RNN2	RNN3
Benchmark		mean	mean	mean	mean
Rediscovery	Calecoxib rediscovery	1.000	1.000	1.000	1.000
	Troglitazone rediscovery	1.000	1.000	1.000	1.000
	Thiothixene rediscovery	1.000	1.000	1.000	1.000
Similarity	Aripiprazole similarity	1.000	1.000	1.000	1.000
	Albuterol similarity	1.000	1.000	1.000	1.000
	Mestranol similarity	0.999	**1.000**	**1.000**	**1.000**
Isomers	C11H24	0.950	**0.983**	**0.984**	**0.987**
	C9H10N2O2PF2CL	0.860	**0.888**	**0.878**	**0.889**
Median	Median molecules 1	0.415	**0.420**	**0.416**	**0.424**
	Median molecules 2	0.407	**0.418**	**0.416**	**0.413**
MPO	Osimertnib MPO	0.897	**0.902**	**0.902**	**0.901**
	Fexofenadine MPO	0.924	**0.949**	**0.937**	**0.949**
	Ranolazine MPO	0.833	**0.839**	**0.845**	**0.847**
	Perindopril MPO	0.766	**0.806**	**0.809**	**0.815**
	Amlodipine MPO	0.888	**0.892**	**0.894**	**0.893**
	Sitgliptin MPO	0.547	0.547	0.529	0.524
	Zaleplon MPO	0.593	**0.605**	**0.607**	**0.600**
	Valsartan SMARTS	0.933	**0.950**	**0.937**	**0.954**
Hopping	decorator hop	0.996	**0.997**	**0.997**	0.994
	scaffold hop	0.994	**0.998**	**0.996**	**0.998**
	Total	17.002	**17.193**	**17.147**	**17.188**

BRICS Enumeration

The next set of results investigated BRICS enumeration as a form of data augmentation for the primary dataset. Table 6-2 shows the goal-directed generation performance of the RNN-LSTM pretrained on the datasets augmented with BRICS enumeration.

The augmented datasets were generated by randomly selecting 0.2%, 1%, and 10% of the original GuacaMol training datasets, i.e. 3000, 12000, and 120000 molecules, respectively. BRICS enumeration was then performed for each SMILES in the training subset. A maximum of 5 enumerated molecules were then randomly selected and added to the overall dataset for augmentation. RNN4-3k is the model where a 0.2%/3k subset of the original GuacaMol dataset were enumerated which resulted in the addition of 2938 molecules to the overall dataset. RNN4-12k is the model where a 1%/12k subset of the original GuacaMol dataset was enumerated which resulted in the addition of 57232 molecules to the overall dataset. Finally, RNN4-120k is the model where a 10%/120k subset of the original GuacaMol dataset was enumerated which resulted in the addition of 572057 molecules in the overall dataset. Five RNN-LSTM models were trained on each of these datasets and their goal-directed generation performances were averaged across the repeats.

Results indicate that BRICS enumeration generally improved total model performance compared to the baseline without data augmentation, with the RNN4-12k model performing the best overall (total mean = 17.258). Interestingly, introducing more enumerated samples to augment the pretraining dataset decreased model performance, i.e. the RNN4-120k has a reduced performance compared to the RNN4-3k and RNN4-12k models. A potential explanation is that enumerated SMILES may have a negative impact on learning the SMILES grammar that are drug-like. This may lead to the observed degradation when optimising for GuacaMol's goal-directed generation tasks.

Table 6-2. Goal-Directed Generation Scores for the BRICS Enumerated Data Augmentation. Bold – greater than baseline, Red – lesser than baseline. Means are calculated from 5 repeats. GuacaMol training dataset size = 1,273,104. Models: Baseline (1.2M); RNN4: Baseline dataset (1.2M) + BRICS enumerated 3k (generated +2938), 12k (generated +57232), and 120k (generated +572057).

		Baseline	RNN4-3k	RNN4-12k	RNN4-120k
Benchmark		mean	mean	mean	mean
Rediscovery	Calecoxib rediscovery	1.000	1.000	1.000	1.000
	Troglitazone rediscovery	1.000	1.000	1.000	1.000
	Thiothixene rediscovery	1.000	1.000	1.000	1.000
Similarity	Aripiprazole similarity	1.000	1.000	1.000	1.000
	Albuterol similarity	1.000	1.000	1.000	1.000
	Mestranol similarity	0.999	**1.000**	**1.000**	**1.000**
Isomers	C11H24	0.950	**0.980**	**0.976**	**0.982**
	C9H10N2O2PF2CL	0.860	**0.862**	**0.904**	**0.882**
Median	Median molecules 1	0.415	0.411	**0.424**	0.399
	Median molecules 2	0.407	**0.417**	**0.417**	0.404
MPO	Osimertnib MPO	0.897	**0.905**	**0.909**	**0.903**
	Fexofenadine MPO	0.924	**0.938**	**0.949**	**0.952**
	Ranolazine MPO	0.833	0.830	**0.845**	**0.850**
	Perindopril MPO	0.766	**0.814**	**0.819**	**0.802**
	Amlodipine MPO	0.888	**0.891**	**0.896**	0.883
	Sitgliptin MPO	0.547	0.533	0.524	0.534
	Zaleplon MPO	0.593	**0.627**	**0.644**	**0.608**
	Valsartan SMARTS	0.933	**0.962**	**0.958**	**0.950**
Hopping	decorator hop	0.996	**1.000**	**0.997**	**0.997**
	scaffold hop	0.994	**0.999**	**0.997**	0.992
	Total	17.002	**17.168**	**17.258**	**17.137**

6.3.2 Celecoxib Rediscovery Trajectory

The score trajectory of the Celecoxib Rediscovery task was investigated to assess how the augmentation of the primary training dataset affects the model's efficiency, i.e. generating high scoring molecules at earlier epochs.

Figure 6-3 shows the trajectories of the RNN-LSTM models pretrained on different primary training datasets. RNN0 indicates the model pretrained on the original GuacaMol training dataset (1.2M). RNN1 indicates the model trained on the randomised SMILES representation rather than the canonical SMILES representation (1.2M). RNN2 indicates the model trained on an augmented dataset, where a single randomised SMILES representation has been generated for each of the baseline samples and included in the overall dataset (2.4M). Finally, RNN3 indicates the model being trained on an augmented dataset, where each of the SMILES from the baseline has been copied once and included in the overall dataset (2.4M). Five RNN-LSTM models were trained on each of these datasets and their goal-directed generation performance were averaged across the repeats. For each model, the averaged training loss, averaged validation loss, and the averaged score of the top-1 molecule at each iteration were plotted.

Results indicate that the duplicated dataset RNN3 rediscovered celecoxib the earliest (at epoch 6), followed by the x1-fold randomised dataset RNN2 (epoch 7), the randomised dataset RNN1 (epoch 8), and finally the baseline dataset RNN0 (epoch 9). These results suggest that models that were trained with data augmentation solved the optimisation problem at an earlier epoch compared to the baseline. Therefore, data augmentation is beneficial for model efficiency.

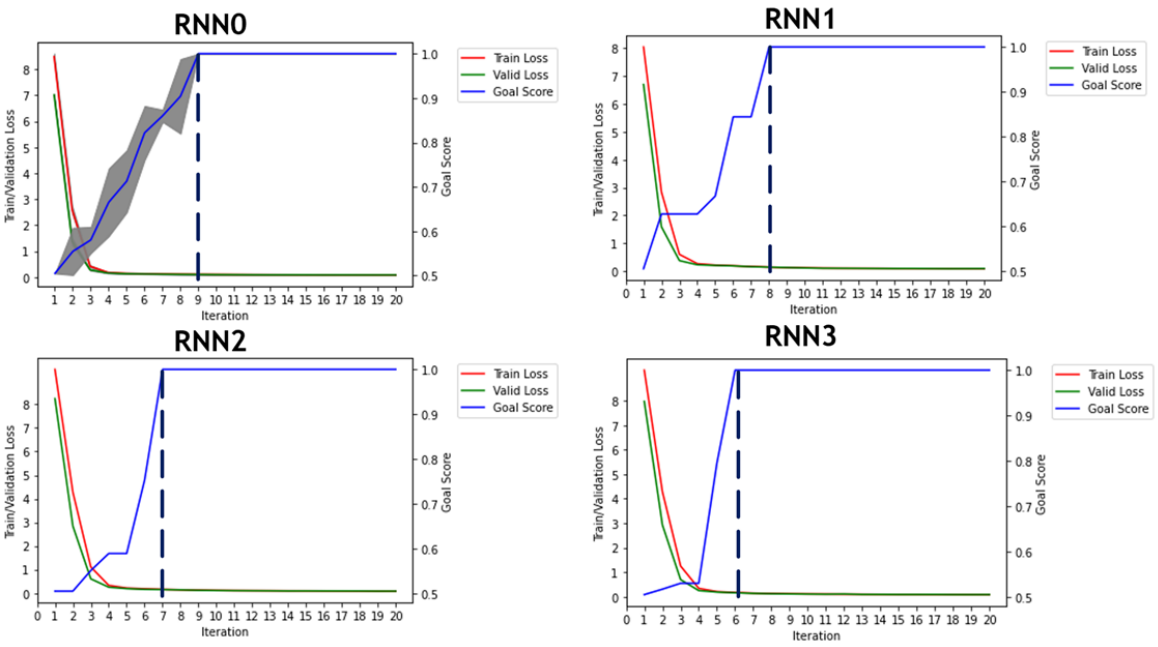


Figure 6-3. The Trajectories of the Celecoxib Rediscovery Task During the Hill-Climbing Iterations of the Different Data Augmented RNNs. Models: RNN0: Baseline (1.2M); RNN1: Replace all canonicalised SMILES with randomised SMILES (1.2M); RNN2: Baseline dataset + randomised SMILES from RNN1 (total 2.4M), RNN3: Baseline dataset + Baseline dataset (total 2.4M). The grey band indicates the standard deviation. The dashed line indicates the iteration at which the optimal solution was found.

6.4 Conclusion

The aim of this chapter was to investigate how data augmentation affects the goal-directed generation performance of generative *de novo* design models. Results indicate that data augmentation has a beneficial effect on model performance, both in terms of the scores of the generated molecules and the efficiency with which the optimal molecules can be generated. Additionally, it may indicate that the ideal pretraining dataset used for a given generative model may be different for different *de novo* design tasks. Therefore, the primary training dataset used to pretrain a generative *de novo* design model should be carefully considered.

The results in this chapter have shown that the different GuacaMol tasks have different difficulty. The rediscovery and similarity tasks can be considered easy as the baseline RNN-LSTM is able to fully optimise the score. In contrast, the MPO benchmark tasks are the most difficult for the baseline RNN-LSTM, indicated by lower scores. Introducing any of the data augmentation operators provided a beneficial effect on model performance. An exception is BRICS enumeration, where there are multiple instances of reduced model performance.

Finally, the results indicate that the Sitagliptin MPO is the most difficult task for the generative model to optimise. A potential way to improve model performance on this task may be to develop a novel method of data augmentation for generative *de novo* design and/or by introducing data augmentation on the secondary dataset, i.e. the data used during the iterative fine-tuning step. These potential solutions are explored in subsequent chapters.

Chapter 7. NLP-inspired Operators

7.1 Introduction

This chapter introduces NLP-inspired operators as potential data augmentation operators for generative *de novo* design and investigates the optimal introduction point of data augmentation. The NLP-inspired operators include atom insertion, atom deletion, atom swapping, and fusion (Figure 7-1). The fusion operator is a combination of insertion, deletion, and/or swapping. In NLP, a simple form of augmentation is noising-based methods, where text samples are modified using either swapping, deletion, insertion, substitution, or a combination of these operators. These noising operations have been shown to improve the performance of a GPT-2 model for text generation (Feng et al., 2020). Therefore, these operators were repurposed for SMILES strings. Additionally, the chapter investigates where data augmentation is best applied during the iterative fine-tuning loop of an RNN-LSTM model for goal-directed generation tasks. These points of entry include the sampling stage, the subset stage, and the subset-loop stage.

As discussed in the Literature Review chapter, the majority of prior work in data augmentation has been to include randomised SMILES (different SMILES representations of a given molecule) in the input data rather than to generate modified SMILES that represent new molecules, as studied here. The very recently published work by Brinkmann et al. (2025) is also based on using data augmentation to manipulate SMILES strings to generate novel molecules, however, there are significant differences between that work and the work of this thesis. First, a wider range of NLP-type operators are explored here including: atom-delete; atom-insert; atom-swap; and combinations of these (Brinkmann et al. (2025) considers atom-delete only). Furthermore, constraints are applied on the operators implemented here that aim to ensure the new SMILES represent valid molecules. Moreover, different points of data augmentation are investigated here. Finally, this work is focused on goal-directed generative design rather than distributional learning as explored by Brinkmann et al. (2025).

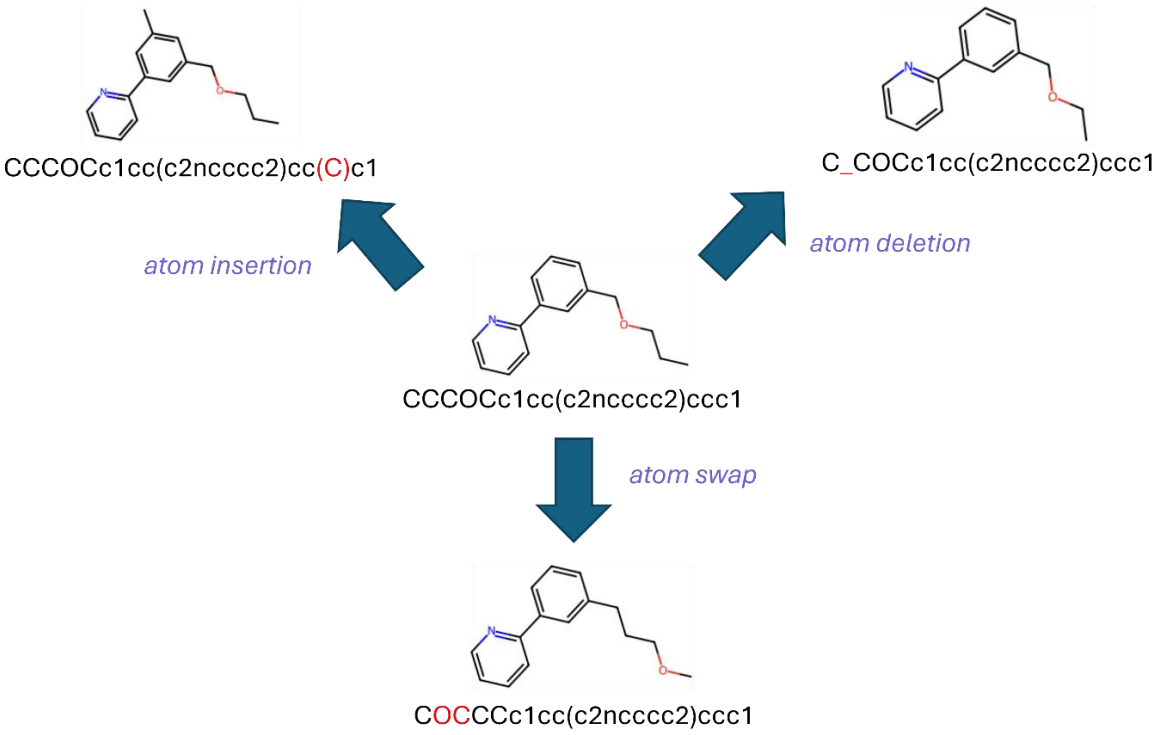


Figure 7-1. NLP-Inspired Operators for Data Augmentation. Atom insertion adds an atom to the SMILES string, atom deletion randomly deletes atoms in a SMILE string, and atom swap randomly swaps atoms within the SMILES string.

The first section of this chapter describes the methods and workflow involved in this chapter. This includes the algorithm that underpins the NLP-inspired operators, how data augmentation may be introduced within the RNN-LSTM for goal-directed generation, and how the methods are evaluated. The next section investigates the extent to which the NLP-inspired operators are able to generate novel and valid SMILES. Then, the operators were applied to the different sections of an RNN-LSTM's iterative-fine tuning loop to identify the optimal point of entry for augmentation. Finally, a parameter search was performed to investigate the best performing augmentation operator.

7.2 Methods

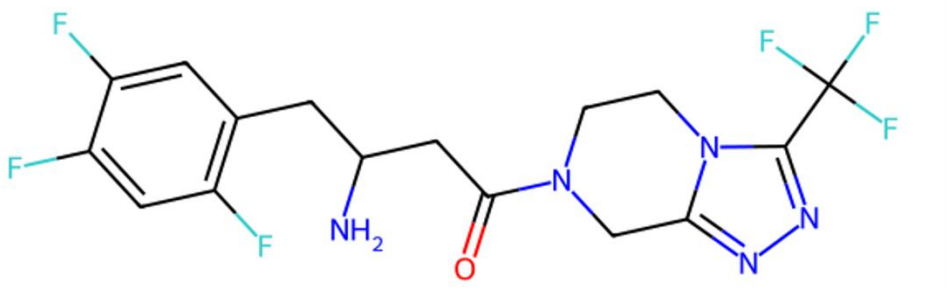
7.2.1 Goal-directed Generation Benchmark Tasks

Goal-directed benchmarks from GuacaMol were used to evaluate model performance, where the was to optimise for specific molecular features or properties. GuacaMol's multi-property optimization (MPO) benchmarks were used to investigate the impact of the NLP-inspired operators on generative *de novo* design models. The RNN-LSTM and Graph-GA methods without data augmentation were used as baseslines to compared the augmented RNN-LSTMs against. The Sitagliptin and Ranolazine MPO benchmark tasks were chosen to evaluate performance as these are tasks where the Graph-GA outperformed the baseline RNN-LSTMs. To explore the effect of the proposed augmentation methods, the top-1, top-10, top-100, and average (top-1, -10, and top-100) MPO scores of the output SMILES were calculated.

Sitagliptin MPO

The Sitagliptin MPO task is to generate molecules that are as dissimilar to sitagliptin while still retaining some of its properties. The baseline RNN-LSTM performed poorly compared to the Graph-GA on this task (Brown *et al.*, 2019).

The MPO is composed of similarity, logP, TPSA, and isomer scoring functions. The similarity between a given SMILES and sitagliptin (Figure 7-2) is calculated using Morgan fingerprints with radius = 2 (corresponding to ECFP4 fingerprints). A gaussian modifier ($\mu = 0$, $\sigma = 0.1$) is then applied to the similarity value for the final output, where higher scores are given to SMILES that are more dissimilar to sitagliptin. The logP and TPSA scoring functions are calculated using RDKit. Gaussian modifiers of ($\mu = 2.0165$, $\sigma = 0.2$) and ($\mu = 77.04$, $\sigma = 5$) are used respectively to target the desired properties, where SMILES that are close to logP = 2.0165 and TPSA = 77.04 are given a higher score. The isomer function gives a higher score to a SMILES that is closer the target formula $C_{16}H_{15}F_6N_5O$. Finally, the contribution of all the components is averaged using the geometric mean.



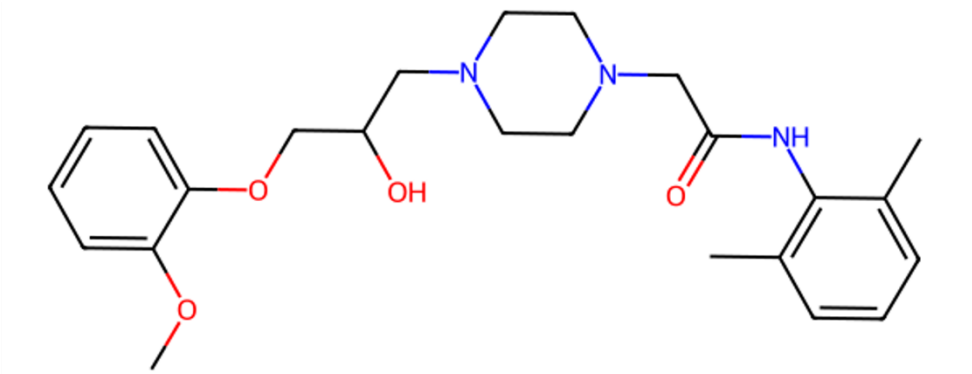
Fc1cc(c(F)cc1F)CC(N)CC(=O)N3Cc2nnc(n2CC3)C(F)(F)F

Figure 7-2. Sitagliptin's Structure and SMILES String Representation.

Ranolazine MPO

The Ranolazine MPO task is to generate molecules that are more polar than the ranolazine molecule whilst adding a single fluorine. Both the baselines RNN-LSTM and Graph-GA are able to generate relatively high scoring molecules for this task. However, the Graph-GA still generates, on average, higher scoring molecules compared to the RNN-LSTM.

The MPO is composed of similarity, logP, TPSA and a fluorine atom count scoring function. The similarity between a given SMILES and ranolazine (Figure 7-3) is calculated using AP fingerprints with a maxLength=10. A threshold modifier (0.7) is then applied to the similarity value for the final output, where the maximum score is given to values above the threshold and a decreasing score is given to values smaller than the threshold. The logP and TPSA scoring functions are calculated using RDKit. MaxGaussian modifiers of ($\mu = 7$, $\sigma = 1$) and ($\mu = 95$, $\sigma = 20$) are used to target the desired properties, where SMILES that have values greater than μ are given a maximum score, and SMILES with values smaller than μ are given a decreasing score. The fluorine atom count scoring function uses a gaussian modifier ($\mu = 1$, $\sigma = 1$) to give a maximum score for SMILES with one fluorine atom. Finally, the contribution of all the components is averaged using the geometric mean.



COc1ccccc1OCC(O)CN2CCN(CC(=O)Nc3c(C)cccc3C)CC2

Figure 7-3. Ranolazine's Structure and SMILES String Representation.

7.2.2 SMILES Atom-level Operators

Three different NLP-inspired operators were explored that modify SMILES at the atom-level. Prior to augmentation processing, encoding and tokenization of SMILES strings are performed (Figure 7-4). SMILES strings are first encoded such that double character elements are represented by a single character. The list of encoding rules is shown in Table 7-1. For example, the SMILES 'CCC(C)(C)Br' is encoded as 'CCC(C)(C)Y'. The SMILES strings were then tokenized into a list of individual characters (these include letters, numbers, and symbols) so that they can be indexed and processed by the NLP-inspired operators. For example, the encoded SMILES 'CCC(C)(C)Y' is tokenized into a list of characters ['C', 'C', 'C', '(', 'C', ')', '(', 'C', ')', 'Y'].

SMILES:
CCC(C)(C)Br

Encoded SMILES:
CCC(C)(C)Y

Tokens:	C	C	C	(	C	)	(	C	)	Y
Index:	0	1	2	3	4	5	6	7	8	9

Figure 7-4. Encoding and Tokenization of SMILES.

The token list can then be augmented using the NLP-inspired operators which are described below, and include swap, delete, insert, and fusion. A common constraint applied for all operators is the keep token constraint. This constraint prevents the operators from making edits on specific tokens as these are likely to lead to invalid SMILES. These token constraints include bonds ('-', '=', '#', and '.'), charges ('-' and '+'), branch indicators ('(' and ')'), ring closure indicators (0-9 and '%'), special atom indicators ('[' and ']') and 'H'. Following the application of an operator, the final encoded output SMILES is then decoded and detokenized to generate the final augmented SMILES output.

Table 7-1. GuacaMol's Encoding Rule. Note that selenium is present in both non-aromatic 'Se' and aromatic form 'se' in the original GuacaMol training dataset and therefore requires separate encoding.

Original Element	Encoded Element
Br	Y
Cl	X
Si	A
Se	Z
se	E

Atom-Swap Operator

The atom-swap operator exchanges the positions of a pair of tokens within a given SMILES string (Figure 7-5). The operator first chooses a pair of random indexes in the tokenized SMILES. The tokens at these positions are then swapped if they pass a series of constraints, which includes: the two indexes are not identical; neither index's token is in the keep tokens list; the tokens are not the same; and neither of the indexes' tokens are aromatic. The aromatic constraint was included as it was observed that swapping aromatic tokens tended to reduce the proportion of augmented SMILES that are valid. If the indexes do not pass the constraints, then another pair of indexes is chosen. The augmented SMILES output can then be fed back into the operator for further augmentation. The number of times a SMILES is fed back into the operator is dictated by the n-repeat parameter.

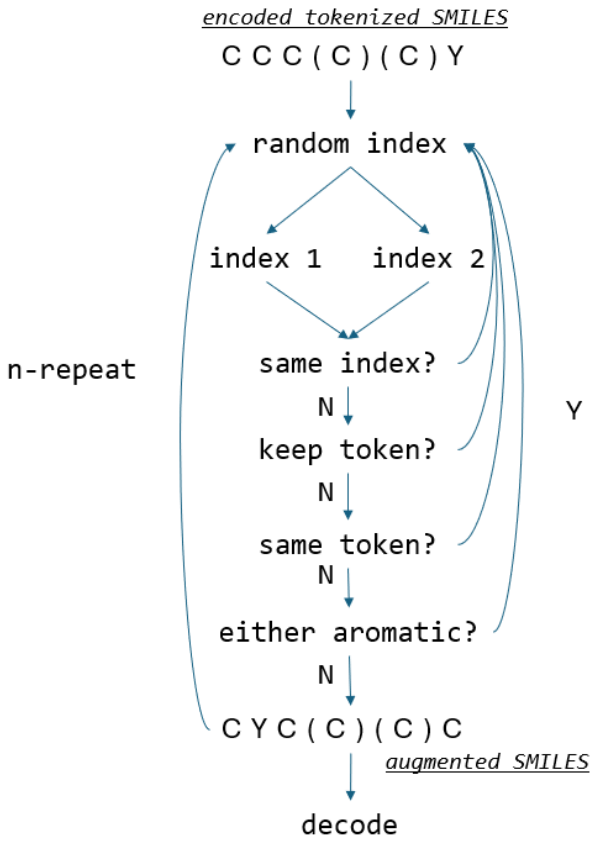


Figure 7-5. The Atom-Swap Operator Algorithm. The operator swaps atoms within a SMILES strings n-times if the chosen indexes pass a series of constraints.

Atom-Delete Operator

The atom-delete operator removes tokens in a SMILES string given a probability *p* (Figure 7-6). The operator iterates through the tokens of the encoded SMILES and for each token, a probability *p* dictates if the token is removed or kept. Constraints included in the operator are a keep token constraint and keeping aromatic tokens. The aromatic constraint is included because it was observed that deleting aromatic tokens reduced the proportion of augmented SMILES that are valid, as above. The augmented SMILES then undergoes a post-processing step to remove unwanted syntax errors. These include removing empty '()', empty '[]', and removing '=' with missing tokens at either side. Note that the post-processing step does not take the keep token list into account.

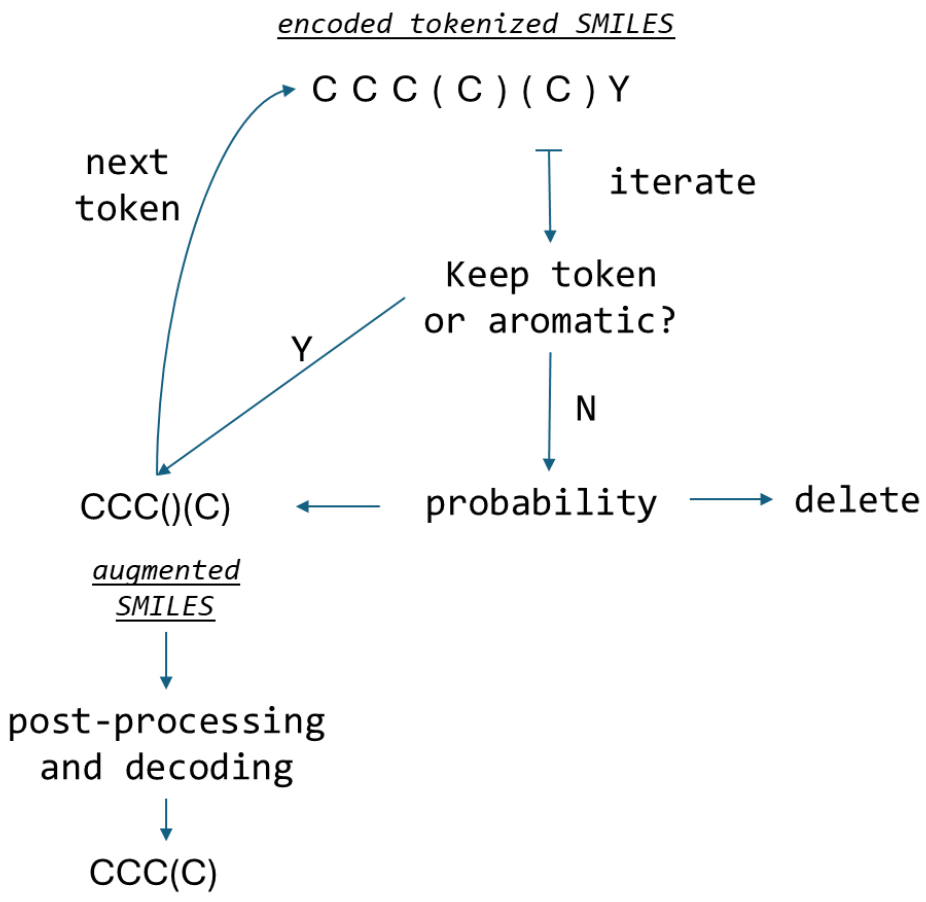


Figure 7-6. The Atom-Delete Operator Algorithm. The operator iterates through a SMILES string and deletes a token with a given probability *p*.

Atom-Insert Operator

The atom-insert operator inserts n tokens into random positions in the SMILES string (Figure 7-7). The operator first chooses a random index from the SMILES string and then a random token from a dictionary of 'C', 'N', 'O', and 'F' (atoms commonly found in organic compounds) is inserted to the right of the index, subject to a constraint. The constraint is such that no insertion is performed if an index's token is a 'keep token'. However, the constraint for this operator is less stringent than the initial keep token constraint discussed above as it does not take the tokens '(', ')', and 'H' into consideration. The assumption here is that inserting atom tokens commonly found in organic compounds to the right of one of these tokens should generally not cause an invalid SMILES string. Furthermore, an aromaticity check is then performed on the index's token. If the token is aromatic, the insert token chosen randomly from the CNOF dictionary is enclosed in parentheses before insertion, so that the atom is added as a substituent on the ring rather than being inserted into the ring.

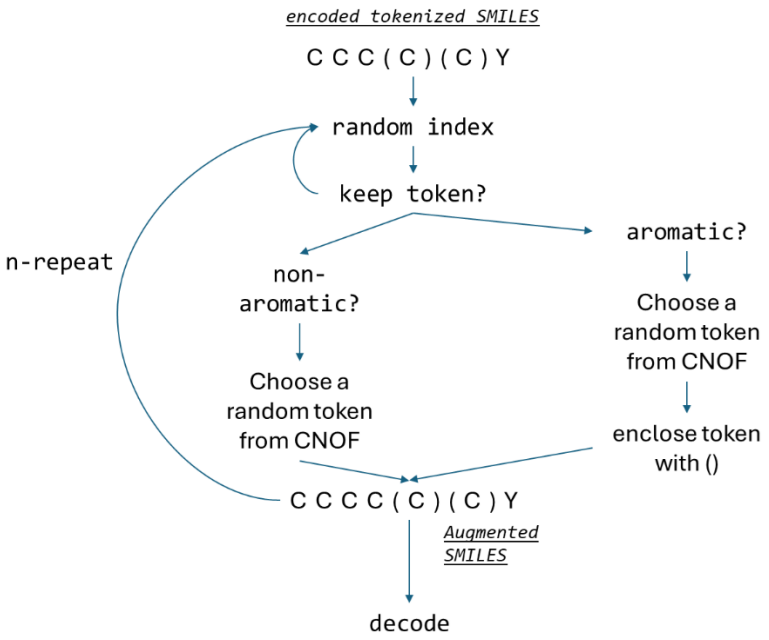


Figure 7-7. The Atom-Insert Operator Algorithm. The operator inserts n random tokens into a SMILES string.

Atom-Fusion Operator

The atom-level fusion, or atom-fusion, operator performs the atom-swap, atom-delete, and atom-insert operations consecutively for a given SMILES string. This means an input SMILES can be processed by more than one operator to produce a single augmented SMILES. There are two different implementations of the atom-fusion which are described below.

The initial atom-fusion operator, termed greedy fusion, has a fixed order of operation with fixed probability and number of executions for each operator. The fixed order is atom-swap, atom-delete, and atom-insert. This order is based on the implementation of Feng et al. (2020) which is from the NLP domain. In the study, the authors performed insertion, deletion, and then swap on the Yelp Reviews text dataset to finetune a pretrained GPT-2 model for text generation tasks. The results indicate that this augmentation improved the quality of the generated texts.

Parameterisation experiments were carried out, described later, to determine the fixed probability/number of executions for each operator. For example, the parameterisation experiments for the Sitagliptin MPO task indicate that the optimal values were atom-swap with n=1, atom-delete with p=0.05, and atom-insert n=3.

The random atom-fusion implementation chooses x number of operators randomly (from atom-swap, atom-delete, atom-insert) with a random magnitude (between 1-6 for n-based operators and 0.05-0.15 for p-based operator) and performs them consecutively for a given SMILES string. The random atom-fusion implementation is based on previous work on automatic data augmentation methods in the computer vision domain, including RandAugment (Cubuk *et al.*, 2019) and TrivialAugment (Müller and Hutter, 2021). RandAugment applies a random number of operations to image-based datasets, such as CIFAR-10/100 and ImageNet, that were used to train deep learning models on classification tasks. Results indicate that RandAugment improved the model's accuracy. On the other hand, the random sampling of the operator's magnitude was based on TrivialAugment (Müller and Hutter, 2021). TrivialAugment performs a uniform random sampling for an augmentation operator's magnitude.

Results indicate that this outperforms methods that performs an extensive parameter search which is resource intensive.

7.2.3 Operator Sense Checks

The success rates of the operators were investigated using the method shown in the flowchart in Figure 7-8. The aim was to identify the extent to which the operators produce unique molecules (not already present in the dataset) and valid molecules. The similarity between the original SMILES and the augmented SMILES output was also calculated. Validity was an important consideration as it is assumed that training with invalid SMILES string will have a negative effect on the model. These metrics are expanded upon below.

In brief, the sense check workflow can be divided into the following steps:

1. Apply the NLP operator for each sample in a dataset of SMILES.
2. The output SMILES then undergoes a series of checks.

   The sense checks categorise the output SMILES into: unaugmented SMILES, i.e., SMILES that has not been augmented; dataset duplicate SMILES, i.e., an augmented SMILES that is already found in the dataset; augmented SMILES, i.e., a SMILES that passes the filters above. Additionally, the augmented SMILES is parsed through RDKit to further categorise it into a valid or invalid augmented SMILES.
3. The final overall metrics are calculated after the full SMILES dataset has been processed. The metrics include the number of samples in the different SMILES categories. Additionally, the similarity and Levenshtein distance of the input SMILES and corresponding valid augmented SMILES are calculated, described in more detail below.

Several metrics were calculated for the operator sense check as described in the workflow above. The percentage validity was calculated as the number of valid SMILES in the augmented SMILES category relative to the number of augmented SMILES. The average similarity between the original SMILES and valid augmented SMILES was calculated using the Tanimoto similarity and Morgan fingerprints (with radius = 2 and nBits=1024) using RDKit. Levenshtein distance, or edit distance, was also calculated. Lavenshtein distance (LD) is the minimum number of single-character edits required to change one string into another string. For example, a Levenshtein distance of 2 indicates that it takes 2 single-character edits to change one string into the other.

An invalid SMILES string is defined as a SMILES strings that cannot be parsed in RDKit. Invalid SMILES categorisation was performed using an adapted code from Skinnider (2024) (original code available at https://doi.org/10.5281/zenodo.10680855).

The error messages that are returned by the implementation include:

- Valence – flagged when an atom's total number of bonds exceeds its maximum valency.
- Aromaticity – flagged when a ring system has an atom that is determined by RDKit to not be a part of a valid ring structure or when there is a kekulization error, i.e., a ring system not forming an alternating single and double bond arrangement.
- Syntax – flagged when there is an unexpected character at an unexpected position that breaks the rules of generating valid SMILES representation.
- Bond exists – flagged when there is an attempt to define an additional bond between two atoms that are already bonded.
- Unclosed ring – flagged when there is a missing ring closure number.
- Parentheses error - flagged when there is an imbalance between the number of opening '(' and closing parentheses ')' used to indicate branching.

A limitation of this current implementation is that it only categorises invalid SMILES according to the first error encountered, i.e. if a SMILES is invalid for more than one of the above reasons, only the first category is recorded.

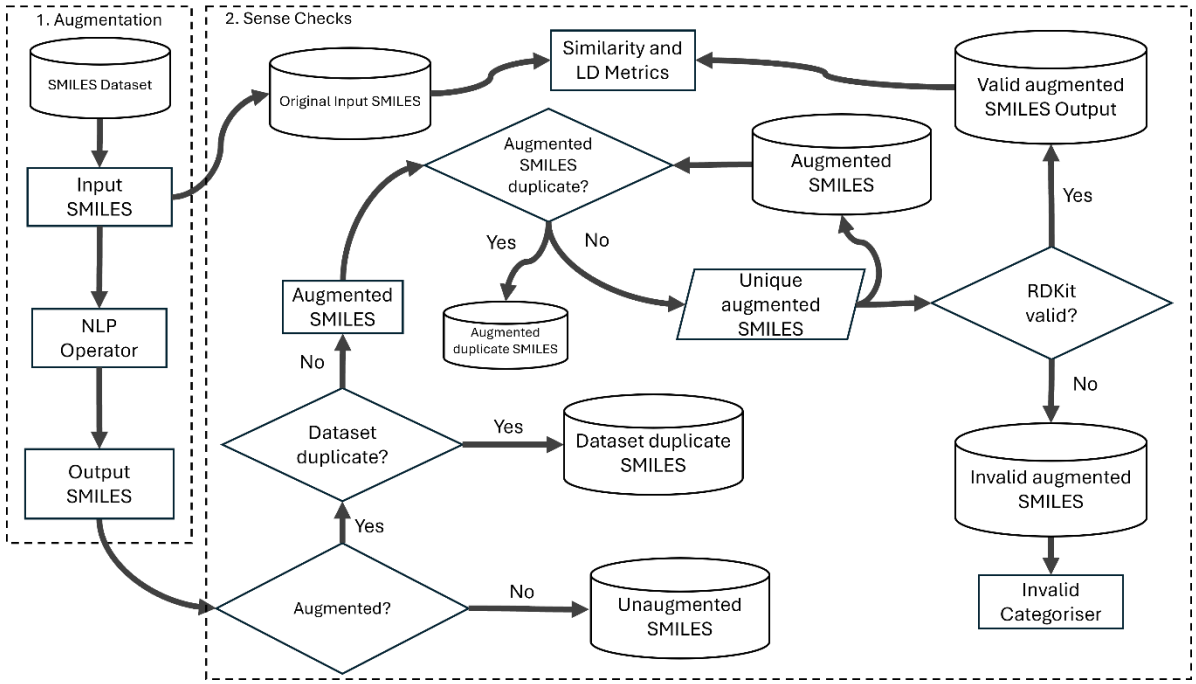


Figure 7-8. NLP-Inspired Operator Sense Check Workflow. The first step of the workflow is to apply the NLP-inspired operators to a SMILES dataset. The next step is to perform sense checks on the output SMILES after the operators have been applied.

Validation of the Operators

The augmentation operators were applied to 1000 SMILES that were selected randomly from the GuacaMol dataset. Different augmentation operators were applied with different magnitudes. This includes the atom-swap operator (with the number of swaps n = 1, 3, and 6), the atom-delete operator (with the probability of deletion p = 0.05, 0.10, and 0.15), and atom-insert (with the number of insertions n = 1, 2, 3). The resulting output SMILES were then categorised and analysed accordingly as shown in the flow chart (Figure 7-8). Each experiment was repeated three times for the same parameters, and the average results are presented.

7.2.4 Augmentation Points

Three different points of data augmentation were explored within the hill-climbing optimisation of the RNN-LSTM (Figure 7-9). The first is at the sample stage, where the SMILES sampled from the model are augmented prior to canonicalisation and scoring.

The second approach is the subset stage, where the top-k SMILES for a given iteration are augmented. This gives an increased number of SMILES that are subsequently used to fine-tune the model. For example, if the top 512 SMILES from the archive are augmented, then up to 1024 SMILES are then used to fine-tune the model (some augmented SMILES will be removed as they are invalid or a duplicate). Randomised SMILES was also used to augment the top k subset for comparison. This approach is inspired by previous works where the top-k SMILES used during reinforcement learning optimisation of generative *de novo* design models were augmented with randomised SMILES (Bjerrum *et al.*, 2023; Guo and Schwaller, 2024).

The final approach, the subset loop stage, is an iterative loop whereby the top-k subset is augmented and scored. The newly created SMILES are then added to the results archive, and a new top-k subset is then selected. The loop is repeated n-times before a final top-k subset is used for fine-tuning the model. The looped subset stage approach is based on the default top-k parameter (i.e. 512). This approach may be seen as a naïve *de novo* design model within the main RNN-LSTM *de novo* design model.

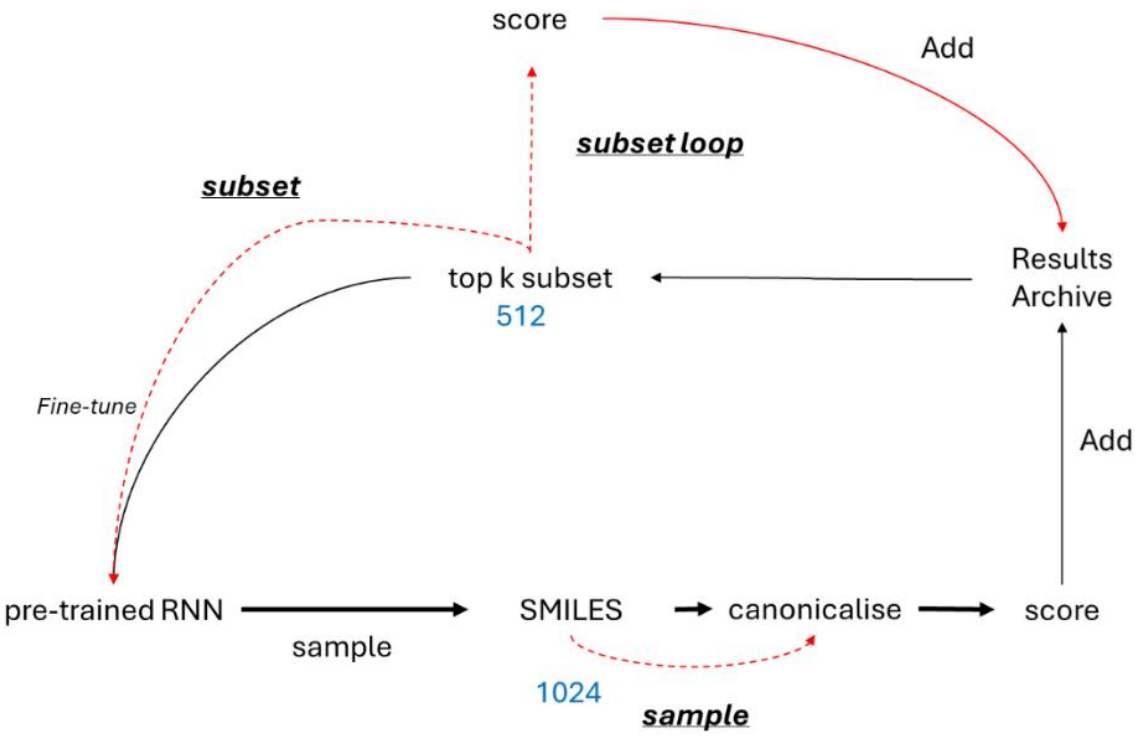


Figure 7-9. The Different Augmentation Points Within the Hill-Climbing Algorithm. Three different potential points of augmentation were explored (red lines) in the fine-tuning optimisation method.

7.3 Results and Discussion

7.3.1 Development of the NLP-inspired Operators

The results shown in this section represent the sense checks performed on the atom-level operators based on the 1000 SMILES selected at random from the GuacaMol dataset.

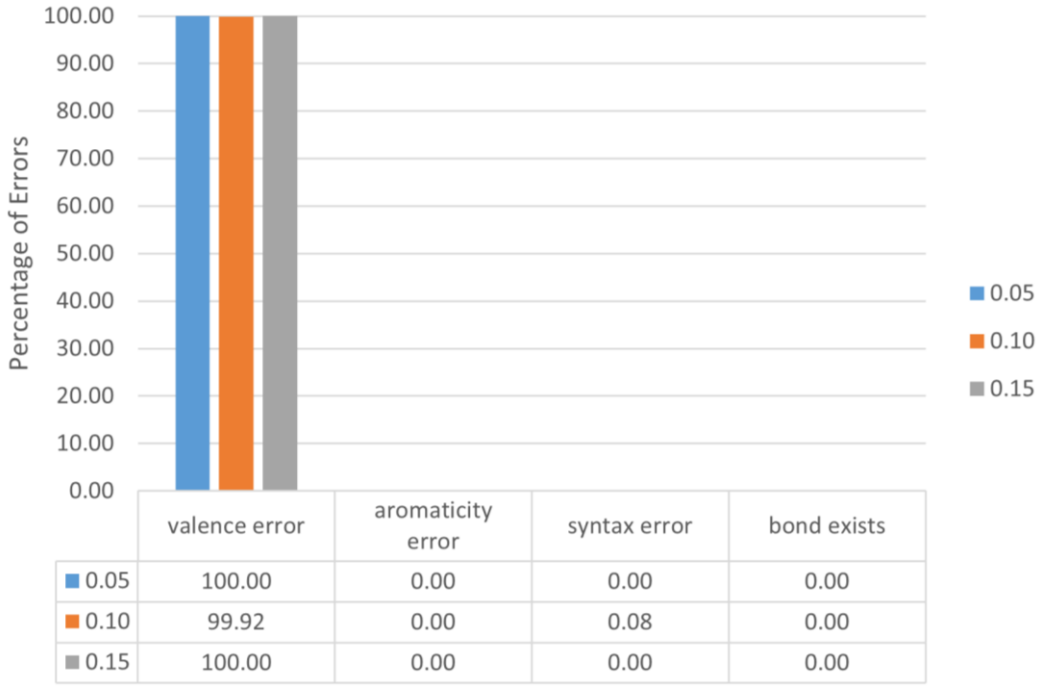
Atom-swap Operator

Table 7-2 shows the sense check results for the atom-swap operator when different numbers (*n* = 1, 3, and 6) of token swaps are made for each SMILES. Total SMILES indicates the input dataset size for the experiment. Augmented SMILES duplicate indicate the number of augmented SMILES already that have already been generated. Augmented SMILES number indicates the number of SMILES that pass the filters above. These SMILES are then further categorised into valid and invalid SMILES. Valid augmented SMILES are compared with their original SMILES and evaluated with Tanimoto similarity and Levenshtein distance (LD).

Results show that the atom-swap operator achieved percentage validities of 62.50% when n = 1, 38.45% when n = 3, and 27.44% when n = 6 (Table 7-2). Categorising the invalid SMILES output indicates that the vast majority of issues were with valency (Figure 7-10). This was likely due to the algorithm not considering the valency of neighbouring atoms. Additionally, a max similarity of 1.0 was also observed. This indicates that the operator has also generated the same molecule but with a different representation, i.e. a randomised SMILES representation. For example, ethanol can be represented both as ‘CCO’ and ‘OCC’. The difference being the positions of C and O have been swapped.

Table 7-2. Atom-Swap Operator Sense Checks. SD indicates the standard deviation of the average Tanimoto similarity calculation. Average LD indicates the average Levenshtein distance. Results shown are averaged across three runs.

Specifications	Total SMILES	Dataset duplicate	Augmented dataset duplicate	Augmented SMILES	Invalid Augmented SMILES	Valid Augmented SMILES	Percentage Valid	Average Similarity	SD Similarity	Max Similarity	Min Similarity	Average LD
1	1000	0.00	0.00	923.50	346.25	577.25	62.50	0.62	0.12	1.00	0.09	2.03
3	1000	0.00	0.00	970.00	597.00	373.00	38.45	0.48	0.14	1.00	0.11	3.70
6	1000	0.00	0.00	980.25	711.25	269.00	27.44	0.42	0.14	0.89	0.11	4.53



	valence error	aromaticity error	syntax error	bond exists
0.05	100.00	0.00	0.00	0.00
0.10	99.92	0.00	0.08	0.00
0.15	100.00	0.00	0.00	0.00

Figure 7-10. Atom-Swap Operator Invalid SMILES Output Categorisation. The y-axis shows the percentage of errors by category.

Atom-delete Operator

The atom-delete operator sense checks investigated performance for different probabilities (p = 0.05, 0.10, and 0.15) of deleting a token as the algorithm iterates through a given SMILES. The operator achieved percentage validities of 76.65% when p = 0.05, 70.78% when p = 0.10, and 60.99% when p = 0.15 (Table 7-3).

Categorising the invalid SMILES output indicates issues in valency, aromaticity, syntax, bond, unclosed ring, and parenthesis error (Figure 7-11). The aromaticity errors were identified to be caused by missing branches or specific atoms within the ring structures of SMILES strings. For example, deleting either symbol '=O' from "CCc1c[nH]c(=O)[nH]c1=O" prevents the SMILES from being kekulized and therefore leads to aromaticity errors.

The syntax errors were identified to be a cause of branch chains not being attached to an atom. For example, the deletion of N in "C(N(CCC))" resulting in "C((CCC))" would lead to an invalid SMILES.

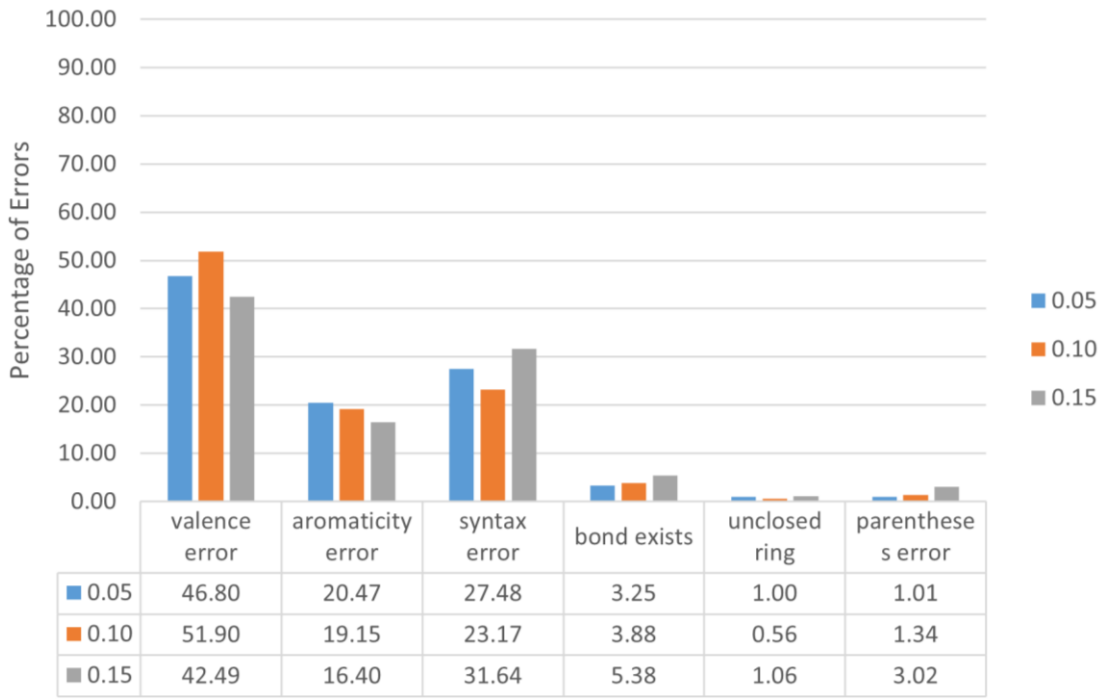
The bond exists error was identified to be due to insufficient atoms between ring closures. This means that the SMILES algorithm is trying to bond atoms together when it should not. For example, the deletion of C within the ring closure 1 of "O=CC1(CO)CO1" to produce "O=CC1(CO)O1" leads to a bond exist error as "ring closure 1 duplicates bond between atom 2 and atom 5".

The unclosed ring error was identified to be caused by SMILES that have a missing branch with a ring closure indicator. For example, when the input 'CC1=CCC2C(C1)c1c(O)cc1OC2' has its '(C1)' branch deleted, it results in an unclosed ring. This can arise if the operator deletes the 'C' and then the post-processing step identifies that the remaining branch '(1)' is invalid and removes it.

Parentheses errors were due to some SMILES missing closing parentheses ')'. For example, the SMILES string 'Cc1cc2cc(OCc(Cl)cc2s1' is missing ')' between the final 'c' and 's' atoms.

Table 7-3. Atom-Delete Operator Sense Checks. SD indicates the standard deviation of the average Tanimoto similarity calculation. Average LD indicates the average Levenshtein distance. Results shown are averaged across three runs.

Specifications	Total SMILES	Dataset Duplicate	Augmented dataset duplicate	Augmented SMILES	Invalid Augmented SMILES	Valid Augmented SMILES	Percentage Valid	Average Similarity	SD Similarity	Max Similarity	Min Similarity	Average LD
0.05	1000	0.00	0.00	574.75	145.50	429.25	74.65	0.72	0.14	1.00	0.17	2.28
0.10	1000	0.00	0.00	768.50	224.50	544.00	70.78	0.67	0.15	1.00	0.21	3.10
0.15	1000	0.00	0.00	863.75	337.00	526.75	60.99	0.62	0.17	1.00	0.09	3.77



	valence error	aromaticity error	syntax error	bond exists	unclosed ring	parentheses error
0.05	46.80	20.47	27.48	3.25	1.00	1.01
0.10	51.90	19.15	23.17	3.88	0.56	1.34
0.15	42.49	16.40	31.64	5.38	1.06	3.02

Figure 7-11. Atom-Delete Operator Invalid SMILES Output Categorisation. The y-axis shows the percentage of errors by category.

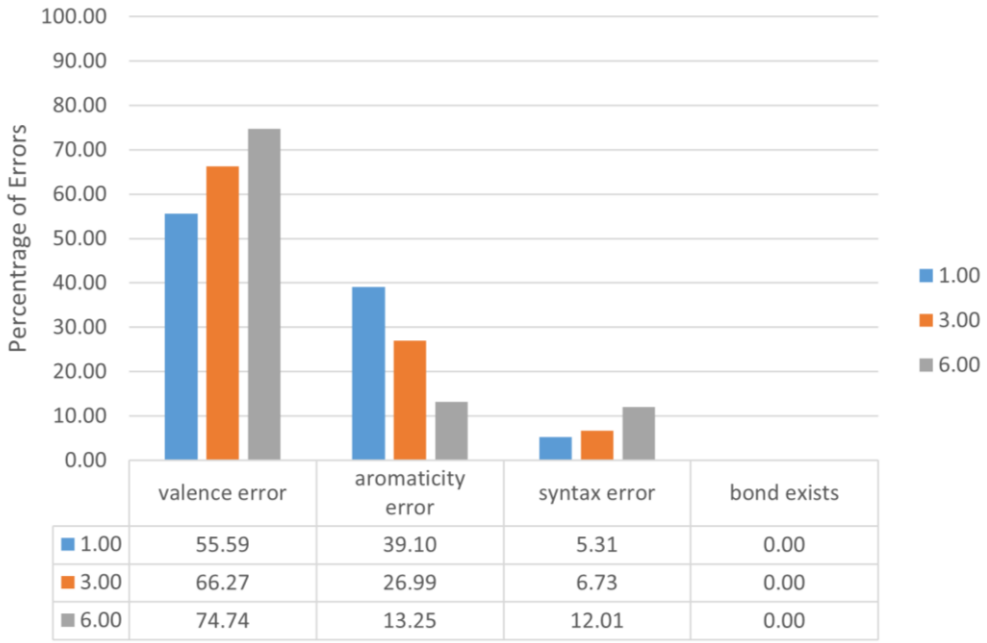
Atom-insert Operator

The atom-insert operator sense checks investigated performance when different numbers (*n* = 1, 3, and 6) of inserts are made for each SMILES. The operator achieved percentage validities of 50.78% when *n* = 1, 24.49% when *n* = 3, and 7.55% when *n* = 6 (Table 7-4).

Categorising the invalid SMILES output indicated issues in valency, aromaticity, and syntax (Figure 7-12). The aromaticity error is likely due to inserting unwanted branches in aromatic rings.

Table 7-4. Atom-Insert Operator Sense Checks. SD indicates the standard deviation of the average Tanimoto similarity calculation. Average LD indicates the average Levenshtein distance. Results shown are averaged across three runs.

Specifications	Total SMILES	Dataset duplicate	Augmented dataset duplicate	Augmented SMILES	Invalid Augmented SMILES	Valid Augmented SMILES	Percentage Valid	Average Similarity	SD Similarity	Max Similarity	Min Similarity	Average LD
1	1000	0.00	157.00	803.00	395.25	407.75	50.78	0.78	0.08	1.00	0.41	2.00
2	1000	0.00	5.75	994.25	750.75	243.50	24.49	0.62	0.14	0.98	0.19	4.39
3	1000	0.00	0.00	1000.00	924.50	75.50	7.55	0.44	0.14	0.84	0.19	8.56



	valence error	aromaticity error	syntax error	bond exists
1.00	55.59	39.10	5.31	0.00
3.00	66.27	26.99	6.73	0.00
6.00	74.74	13.25	12.01	0.00

Figure 7-12. Atom-Insert Operator Invalid SMILES Output Categorisation. Data shown as the percentage of different invalid categories.

Discussion

The results from the sense checks indicate that the main contributing factor for the invalid SMILES generated by the different operators is due to valency errors. The valency errors are due to the algorithms being agnostic to the valency of neighbouring atoms. Additionally, it was not surprising to observe that as the noise, i.e. the magnitude of a given operator, is increased, it is more likely that the output SMILES become invalid.

The subsequent use of the atom-level operators described from this point involves rerunning of the operators until a valid SMILES output is generated. The results from this section have shown that all operators are able to output a valid SMILES and therefore rerunning the operators until a valid SMILES is generated is a possible solution. However, there is a limit of 10 reruns to prevent rare cases where a valid SMILES is not generated.

7.3.2 RNN-LSTM- Augmentation Points

Potential augmentation points in the fine-tuning stage of an RNN-LSTM using hill-climbing optimisation were explored. These points of augmentation include the sample stage, the subset stage, and the subset loop stage (Figure 7-9).

Figure 7-13 and Figure 7-14 shows the mean results of the top-100 SMILES generated across repeats for the baseline models (RNN-LSTM and Graph-GA), i.e., without any data augmentation, as well as the RNN-LSTM augmented at different points (sample, subset, subset-loop) tasked on the Sitagliptin MPO and Ranolazine MPO, respectively. The subset-loop was performed with loop l = 5. The atom-level operators applied included atom-delete, atom-insert, atom-swap, and atom-fusion 3. The atom-delete's probability of deletion p is chosen at random within the range 0.05 and 0.15. The atom-insert's number of insertions n is chosen at random within the range 1-6. The atom-swap's number of swaps n is chosen at random within the range 1-6. The atom-fusion 3 operator randomly selects an atom-level operator from the above and applies it to a given SMILES, and this is repeated for a total of 3 times.

Additionally, randomised SMILES augmentation at the subset stage was explored to compare against the NLP-inspired operators. The amount of data augmentation performed using randomised SMILES N includes 2, 5, and 10. An augmentation with N = 2 indicates that the number of SMILES has been doubled, as each SMILES will have been augmented once, and so on. Augmentation with N2 was performed because this matches the number of augmentations performed when using the atom-level operators. Augmentations with N5 and N10 were also performed because these were found to be the optimal number of augmentations when using randomised SMILES (Bjerrum *et al.*, 2023).

Results for the Sitagliptin MPO task indicate that the application of any of the atom-level operators at any of the augmentation points generally improved the average MPO score of the generated molecules compared to the RNN-LSTM baseline (Figure 7-13 and Appendix Table 0-1). The biggest improvement was observed when performing augmentation with the subset loop. Therefore, augmentation at the

subset loop level will be the focus for the rest of this chapter. However, the Graph-GA model still performs the best for the Sitagliptin MPO task. Also, using randomised SMILES as a form of data augmentation indicated some improvement on the Sitagliptin MPO task, with increasing *N* leading to an increased average score. However, augmenting with an atom-level operator at the subset loop outperforms randomised SMILES (Figure 7-13). This suggests that augmenting using atom-level operators at the subset loop is beneficial for generative *de novo* design.

On the other hand, results for the Ranolazine MPO task indicates that the application of the atom-level operators is best performed at the subset-loop stage (Figure 7-14 and Appendix Table 0-2). At the sample and subset stages, atom-delete and atom-insert tend to improve model performance more than the atom-swap and atom-fusion operators. Additionally, randomised SMILES tend to perform worst compared to the atom-level operators.

Discussion

Results indicate that the best point of augmentation is during the subset loop stage with *I* = 5. It is likely because the atom-level operators have more opportunities to introduce better modifications on the SMILES. Additionally, the atom-level operators tend to outperform randomised SMILES. The subsequent sections will explore the subset loop stage in more detail as it has been identified to improve performance the most for RNN-LSTM training.

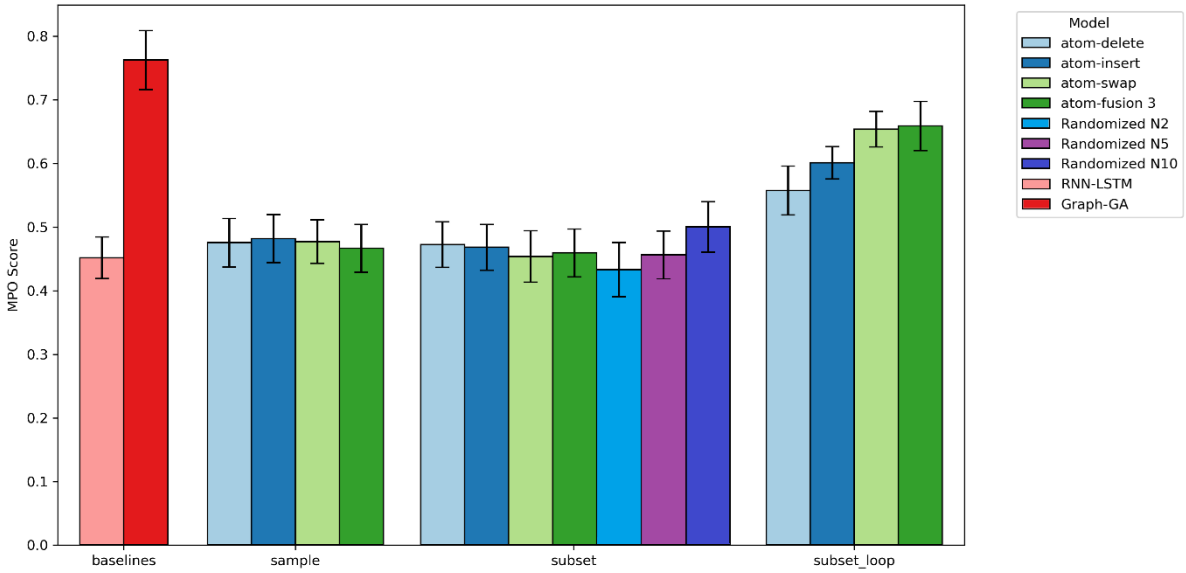


Figure 7-13. Mean Sitagliptin MPO Scores When Augmented SMILES are Introduced at Different Points of the Fine-Tuning Step of the RNN-LSTM. The mean is the average Sitagliptin MPO score of the top-100 SMILES generated across repeats. Each bar is the average across three replicates with standard deviation shown.

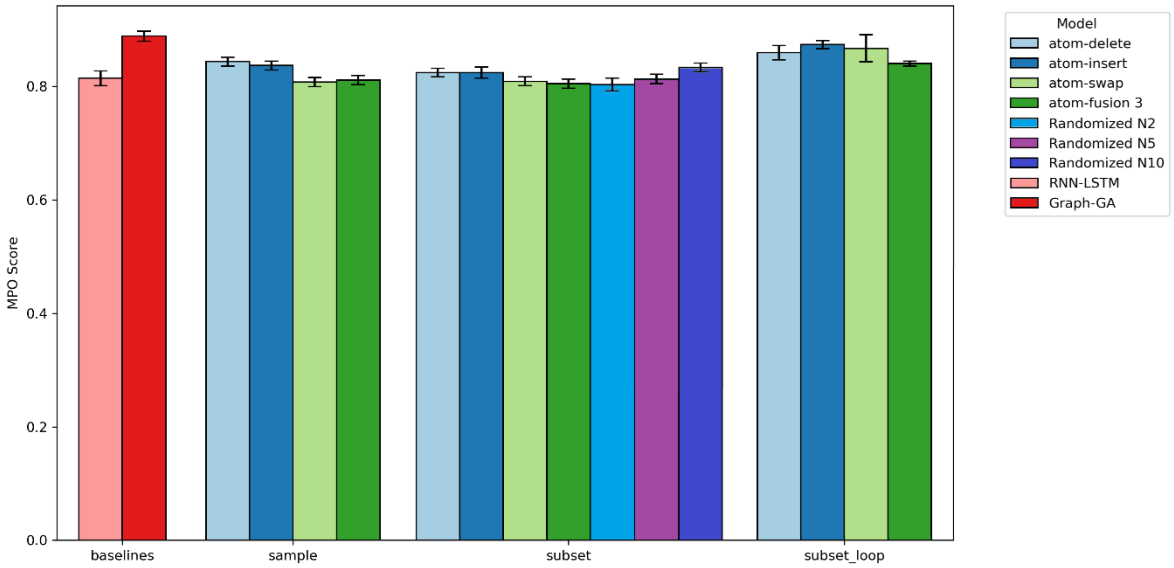


Figure 7-14. Mean Ranolazine MPO Scores When Augmented SMILES are Introduced at Different Points of the Fine-Tuning Step of the RNN-LSTM. The mean is the average Ranolazine MPO score of the top-100 SMILES generated across repeats. Each bar is the average across three replicates with standard deviation shown.

7.3.3 RNN-LSTM - Subset-Loop Augmentation

The aim of these experiment is to investigate the most optimal parameters for augmenting at the subset loop. Parameters explored include the type of atom-level operator, the magnitude of the atom-level operator, and the number of subset loops.

Atom-Swap Operator

Table 7-5 and Table 7-6 show the mean over three repeats of the top-1, top-10, top-100, and the average (top-1, top-10, and top-100) MPO scores, Sitagliptin and Ranolazine tasks, respectively, of the SMILES generated from the baseline and augmented models. The augmentation performed here is at the subset-loop stage with the atom-swap operator.

The results indicate that, for the Sitagliptin MPO task, increasing the number of subset loops *l* generally improved performance with the highest average of 0.718 at swap n = 1 with l = 10 subset loops when compared to the average RNN-LSTM baseline of 0.527 (Table 7-5). The performance of the model with *n* = 1 swaps with *l* = 5 subset loops is close, with an average score of 0.716. Increasing the number of swaps per SMILES, i.e., *n*, within each subset loop reduced the model performance.

On the other hand, for the Ranolazine MPO task, the optimal number of subset loops is *l* = 5 with the highest average of 0.886 at swap *n* = 1. Further increasing the subset loop for Ranolazine MPO led to a general decrease in model performance. Similar to the Sitagliptin MPO task, increasing the number of swaps per SMILES *n* within each subset loop reduced the model performance.

Table 7-5. Model Performance on the Sitagliptin MPO Task When Performing Atom-Level Swap on the Subset Loop Stage of Iterative Fine-Tuning. Results presented are the top-1, top-10, and top-100 MPO scores of the molecules generated. The average column is the mean of the top-1, top-10, and top-100 MPO scores. Each result is the average across three replicates. Brackets indicate the standard deviation. Bold indicates the best augmented result compared to the RNN-LSTM baseline.

	top 1	**top 10**	**top 100**	**average**
RNN-LSTM	0.598 (0.00)	0.530 (0.00)	0.453 (0.00)	0.527 (0.00)
RNN-LSTM Swap n = 1 Subset Loop 1	0.649 (0.02)	0.618 (0.02)	0.572 (0.02)	0.613 (0.02)
RNN-LSTM Swap n = 3 Subset Loop 1	0.641 (0.04)	0.598 (0.01)	0.538 (0.00)	0.592 (0.02)
RNN-LSTM Swap n = 6 Subset Loop 1	0.606 (0.01)	0.560 (0.01)	0.508 (0.00)	0.558 (0.01)
RNN-LSTM Swap n = 1 Subset Loop 5	0.735 (0.05)	**0.720 (0.04)**	0.693 (0.03)	0.716 (0.04)
RNN-LSTM Swap n = 3 Subset Loop 5	0.681 (0.02)	0.647 (0.01)	0.611 (0.00)	0.647 (0.01)
RNN-LSTM Swap n = 6 Subset Loop 5	0.629 (0.02)	0.605 (0.01)	0.558 (0.01)	0.597 (0.01)
RNN-LSTM Swap n = 1 Subset Loop 10	**0.740 (0.07)**	0.715 (0.03)	**0.699 (0.04)**	**0.718 (0.05)**
RNN-LSTM Swap n = 3 Subset Loop 10	0.706 (0.07)	0.692 (0.06)	0.666 (0.06)	0.688 (0.06)
RNN-LSTM Swap n = 6 Subset Loop 10	0.670 (0.01)	0.646 (0.01)	0.608 (0.01)	0.641 (0.00)
Graph-GA	0.806 (0.02)	0.790 (0.05)	0.763 (0.05)	0.786 (0.04)

Table 7-6. Model Performance on the Ranolazine MPO Task When Performing Atom-Level Swap on the Subset Loop Stage of Iterative Fine-Tuning. Results presented are the top-1, top-10, and top-100 MPO scores of the molecules generated. The average column is the mean of the top-1, top-10, and top-100 MPO scores. Each result is the average across three replicates. Brackets indicate the standard deviation. Bold indicates the best augmented result compared to the RNN-LSTM baseline.

	top 1	**top 10**	**top 100**	**average**
RNN-LSTM	0.847 (0.01)	0.835 (0.01)	0.814 (0.01)	0.832 (0.01)
RNN-LSTM Swap n = 1 Subset Loop 1	0.854 (0.01)	0.849 (0.01)	0.839 (0.02)	0.847 (0.01)
RNN-LSTM Swap n = 3 Subset Loop 1	0.848 (0.00)	0.836 (0.01)	0.821 (0.00)	0.835 (0.00)
RNN-LSTM Swap n = 6 Subset Loop 1	0.846 (0.02)	0.833 (0.01)	0.817 (0.00)	0.832 (0.01)
RNN-LSTM Swap n = 1 Subset Loop 5	**0.890 (0.01)**	**0.886 (0.01)**	**0.881 (0.01)**	**0.886 (0.01)**
RNN-LSTM Swap n = 3 Subset Loop 5	0.856 (0.01)	0.852 (0.01)	0.846 (0.01)	0.852 (0.01)
RNN-LSTM Swap n = 6 Subset Loop 5	0.863 (0.01)	0.859 (0.02)	0.851 (0.02)	0.858 (0.02)
RNN-LSTM Swap n = 1 Subset Loop 10	0.878 (0.01)	0.876 (0.01)	0.872 (0.01)	0.875 (0.01)
RNN-LSTM Swap n = 3 Subset Loop 10	0.872 (0.01)	0.871 (0.01)	0.866 (0.01)	0.870 (0.01)
RNN-LSTM Swap n = 6 Subset Loop 10	0.860 (0.01)	0.857 (0.01)	0.853 (0.01)	0.856 (0.01)
Graph-GA	0.899 (0.01)	0.896 (0.01)	0.888 (0.01)	0.894 (0.01)

Atom-Delete Operator

Table 7-7 and Table 7-8 show the mean over three repeats of the top-1, top-10, top-100, and the average (top-1, top-10, and top-100) MPO scores, Sitagliptin and Ranolazine tasks, respectively, of the SMILES generated from the baseline and augmented models. The augmentation performed here is at the subset-loop stage with the atom-delete operator. Results indicate that a subset loop with *l* = 10 performed the best for both Sitagliptin MPO and Ranolazine MPO. With the highest average of 0.647 with *p* = 0.05 and 0.874 with *p* = 0.05. Additionally, increasing the probability of token deletion *p* within each of the subset loops generally reduced the model performance across all metrics. An exception is for the subset l = 1 with the Sitagliptin MPO task, where model performance increases as noise is increased. This may suggest that only one loop may not be enough to introduce variation into the augmented SMILES.

Table 7-7. Model Performance on the Sitagliptin MPO Task When Performing Atom-Level Delete on the Subset Loop Stage of Iterative Fine-Tuning. Results presented are the top-1, top-10, and top-100 MPO scores of the molecules generated. The average column is the mean of the top-1, top-10, and top-100 MPO scores. Each result is the average across three replicates. Brackets indicate the standard deviation. Bold indicates the best augmented result compared to the RNN-LSTM baseline.

	top 1	**top 10**	**top 100**	**average**
RNN-LSTM	0.598 (0.00)	0.530 (0.00)	0.453 (0.00)	0.527 (0.00)
RNN-LSTM Del p = 0.05 Subset Loop 1	0.619 (0.03)	0.577 (0.01)	0.513 (0.01)	0.570 (0.02)
RNN-LSTM Del p = 0.10 Subset Loop 1	0.621 (0.04)	0.576 (0.02)	0.512 (0.00)	0.570 (0.02)
RNN-LSTM Del p = 0.15 Subset Loop 1	0.640 (0.03)	0.587 (0.01)	0.520 (0.01)	0.582 (0.02)
RNN-LSTM Del p = 0.05 Subset Loop 5	0.694 (0.07)	0.620 (0.02)	0.561 (0.01)	0.625 (0.03)
RNN-LSTM Del p = 0.10 Subset Loop 5	0.667 (0.08)	0.620 (0.04)	0.564 (0.02)	0.617 (0.05)
RNN-LSTM Del p = 0.15 Subset Loop 5	0.645 (0.04)	0.604 (0.02)	0.559 (0.01)	0.603 (0.02)
RNN-LSTM Del p = 0.05 Subset Loop 10	**0.719 (0.04)**	0.642 (0.01)	0.579 (0.01)	**0.647 (0.02)**
RNN-LSTM Del p = 0.10 Subset Loop 10	0.684 (0.02)	**0.652 (0.03)**	0.591 (0.02)	0.643 (0.02)
RNN-LSTM Del p = 0.15 Subset Loop 10	0.684 (0.07)	**0.652 (0.07)**	**0.598 (0.06)**	0.645 (0.07)
Graph-GA	0.806 (0.02)	0.790 (0.05)	0.763 (0.05)	0.786 (0.04)

Table 7-8. Model Performance on the Ranolazine MPO Task When Performing Atom-Level Delete on the Subset Loop Stage of Iterative Fine-Tuning. Results presented are the top-1, top-10, and top-100 MPO scores of the molecules generated. The average column is the mean of the top-1, top-10, and top-100 MPO scores. Each result is the average across three replicates. Brackets indicate the standard deviation. Bold indicates the best augmented result compared to the RNN-LSTM baseline.

	top 1	**top 10**	**top 100**	**average**
RNN-LSTM	0.847 (0.01)	0.835 (0.01)	0.814 (0.01)	0.832 (0.01)
RNN-LSTM Del p = 0.05 Subset Loop 1	0.862 (0.01)	0.852 (0.01)	0.838 (0.01)	0.851 (0.01)
RNN-LSTM Del p = 0.10 Subset Loop 1	0.854 (0.01)	0.842 (0.01)	0.829 (0.01)	0.842 (0.01)
RNN-LSTM Del p = 0.15 Subset Loop 1	0.849 (0.01)	0.839 (0.01)	0.822 (0.01)	0.837 (0.01)
RNN-LSTM Del p = 0.05 Subset Loop 5	0.874 (0.02)	0.868 (0.02)	0.861 (0.02)	0.867 (0.02)
RNN-LSTM Del p = 0.10 Subset Loop 5	0.858 (0.01)	0.854 (0.01)	0.848 (0.01)	0.853 (0.01)
RNN-LSTM Del p = 0.15 Subset Loop 5	0.841 (0.00)	0.838 (0.00)	0.834 (0.00)	0.838 (0.00)
RNN-LSTM Del p = 0.05 Subset Loop 10	**0.877 (0.03)**	**0.875 (0.03)**	**0.869 (0.03)**	**0.874 (0.03)**
RNN-LSTM Del p = 0.10 Subset Loop 10	0.862 (0.04)	0.856 (0.04)	0.849 (0.03)	0.855 (0.04)
RNN-LSTM Del p = 0.15 Subset Loop 10	0.856 (0.02)	0.845 (0.01)	0.838 (0.02)	0.847 (0.02)
Graph-GA	0.899 (0.01)	0.896 (0.01)	0.888 (0.01)	0.894 (0.01)

Atom-Insert Operator

Table 7-9 and Table 7-10 show the mean top-1, top-10, top-100, and the average (top-1, top-10, and top-100) MPO scores, Sitagliptin and Ranolazine tasks, respectively, of the SMILES generated from the baseline and augmented models. The augmentation performed here is at the subset-loop stage with the atom-insert operator. Results for both MPO tasks suggest that a subset loop with *l* = 10 performs the best. The highest average of 0.723 is achieved with *n* = 3 for Sitagliptin MPO and 0.898 with *n* = 1 for Ranolazine MPO. Additionally, the RNN-LSTM with atom-insert *n* = 1 at subset loop *l* = 10 shows a minor improvement when compared to the Graph-GA, average = 0.898 (0.01) vs average = 0.894 (0.01) respectively. This may be because the atom-insert operator increases the likelihood to insert F atoms, which is one of the scoring functions of the Ranolazine MPO.

Table 7-9. Model Performance on the Sitagliptin MPO Task When Performing Atom-Level Insert on the Subset Loop Stage of Iterative Fine-Tuning. Results presented are the top-1, top-10, and top-100 MPO scores of the molecules generated. The average column is the mean of the top-1, top-10, and top-100 MPO scores. Each result is the average across three replicates. Brackets indicate the standard deviation. Bold indicates the best augmented result compared to the RNN-LSTM baseline.

	top 1	**top 10**	**top 100**	**average**
RNN-LSTM	0.598 (0.00)	0.530 (0.00)	0.453 (0.00)	0.527 (0.00)
RNN-LSTM Ins n = 1 Subset Loop 1	0.636 (0.01)	0.605 (0.02)	0.542 (0.00)	0.595 (0.01)
RNN-LSTM Ins n = 3 Subset Loop 1	0.674 (0.02)	0.607 (0.03)	0.513 (0.01)	0.598 (0.02)
RNN-LSTM Ins n = 6 Subset Loop 1	0.623 (0.03)	0.550 (0.00)	0.462 (0.01)	0.545 (0.01)
RNN-LSTM Ins n = 1 Subset Loop 5	0.701 (0.02)	0.681 (0.02)	0.641 (0.02)	0.675 (0.02)
RNN-LSTM Ins n = 3 Subset Loop 5	0.719 (0.04)	0.701 (0.05)	0.643 (0.04)	0.688 (0.04)
RNN-LSTM Ins n = 6 Subset Loop 5	0.629 (0.04)	0.583 (0.03)	0.488 (0.00)	0.567 (0.02)
RNN-LSTM Ins n = 1 Subset Loop 10	0.692 (0.01)	0.681 (0.01)	0.649 (0.02)	0.674 (0.01)
RNN-LSTM Ins n = 3 Subset Loop 10	**0.770 (0.03)**	**0.727 (0.03)**	**0.671 (0.04)**	**0.723 (0.03)**
RNN-LSTM Ins n = 6 Subset Loop 10	0.628 (0.03)	0.600 (0.03)	0.525 (0.05)	0.584 (0.04)
Graph-GA	0.806 (0.02)	0.790 (0.05)	0.763 (0.05)	0.786 (0.04)

Table 7-10. Model Performance on the Ranolazine MPO Task When Performing Atom-Level Insert on the Subset Loop Stage of Iterative Fine-Tuning. Results presented are the top-1, top-10, and top-100 MPO scores of the molecules generated. The average column is the mean of the top-1, top-10, and top-100 MPO scores. Each result is the average across three replicates. Brackets indicate the standard deviation. Bold indicates the best augmented result compared to the RNN-LSTM baseline.

	top 1	**top 10**	**top 100**	**average**
RNN-LSTM	0.847 (0.01)	0.835 (0.01)	0.814 (0.01)	0.832 (0.01)
RNN-LSTM Ins n = 1 Subset Loop 1	0.850 (0.01)	0.844 (0.01)	0.833 (0.01)	0.842 (0.01)
RNN-LSTM Ins n = 3 Subset Loop 1	0.855 (0.00)	0.847 (0.00)	0.832 (0.01)	0.845 (0.01)
RNN-LSTM Ins n = 6 Subset Loop 1	0.846 (0.00)	0.828 (0.01)	0.806 (0.00)	0.827 (0.00)
RNN-LSTM Ins n = 1 Subset Loop 5	0.892 (0.02)	0.888 (0.02)	0.884 (0.01)	0.888 (0.02)
RNN-LSTM Ins n = 3 Subset Loop 5	0.879 (0.02)	0.876 (0.02)	0.871 (0.02)	0.875 (0.02)
RNN-LSTM Ins n = 6 Subset Loop 5	0.849 (0.01)	0.840 (0.01)	0.829 (0.01)	0.839 (0.01)
RNN-LSTM Ins n = 1 Subset Loop 10	**0.900 (0.01)**	**0.898 (0.01)**	**0.895 (0.01)**	**0.898 (0.01)**
RNN-LSTM Ins n = 3 Subset Loop 10	0.889 (0.02)	0.886 (0.02)	0.878 (0.02)	0.884 (0.02)
RNN-LSTM Ins n = 6 Subset Loop 10	0.870 (0.01)	0.859 (0.02)	0.850 (0.02)	0.860 (0.02)
Graph-GA	0.899 (0.01)	0.896 (0.01)	0.888 (0.01)	0.894 (0.01)

Discussion

In general, it was observed that performing more subset loops improved overall model performance. This is likely because it provides more opportunity for the operators to generate higher scoring SMILES for the model to learn from. However, this come at higher computational costs as more scoring function calls are required.

For the Sitagliptin MPO task, the best performing individual atom-level operator is the atom-insert. The atom-insert operator generated the highest scoring top-1 SMILES (0.770) compared to the previous operators where the highest scoring generated SMILES achieved a score of 0.740 (atom-swap) and 0.719 (atom-delete). In terms of the average score, atom-insert generated the highest scoring average (0.723) compared to atom-swap (0.718) and atom-delete (0.647).

For the Ranolazine MPO task, the best performing individual atom-level operator is also the atom-insert. The atom-insert operator generated the highest scoring top-1 SMILES (0.900) compared to the previous operators where the highest generated SMILES achieved a score of 0.890 (atom-swap) and 0.877 (atom-delete). For the average score, atom-insert also generated the highest scoring average (0.898) compared to atom-swap (0.886) and atom-delete (0.874)

The atom-insert operator is likely to be the most beneficial on the sitagliptin MPO task because it is introducing F atoms into the SMILES, which produces augmented SMILES that score highly on sitagliptin MPO's $C_{16}H_{15}F_6N_5O$ isomer scoring function and ranolazine MPO's F atom count scoring function.

7.3.4 RNN-LSTM Subset-Loop Augmentation with Greedy Atom-Fusion

This section investigated the application of atom-fusion augmentation on RNN-LSTM performance on the Sitagliptin MPO and Ranolazine MPO tasks. The parameters used for each of the atom-level operator was informed by the results of the previous section.

For the sitagliptin MPO, the atom-fusion parameters were atom-swap with n = 1, atom-delete with p = *0.05*, and atom-insert with n = 3. Table 7-11 shows the result for the RNN-LSTM augmented at the subset-loop stage with the greedy atom-fusion operator for the Sitagliptin MPO task. Results indicate that increasing the number of subset loops improved performance when compared to the RNN-LSTM baseline. Additionally, the subset loop 10 also marginally outperformed the Graph-GA in the top-1 generated molecule (0.807 and 0.806 respectively) and performed closely for the top-10 generated molecules (0.789 and 0.790 respectively. However, the method did not perform as closely at the top-100 metric.

On the other hand, for the Ranolazine MPO task, the atom-fusion parameters were atom-level operators: atom-swap n = 1, atom-delete p = *0.05*, atom-insert n = 1. Table 7-12 shows the result for the RNN-LSTM augmented at the subset-loop stage with the greedy atom-fusion operator for the Ranolazine MPO task. Like the greedy atom-fusion results for the Sitagliptin MPO task, results indicate that increasing the number of subset loops improved model performance. However, the greedy atom-fusion operator did not outperform the Graph-GA in any of the metrics for this task.

The results indicate that the atom-fusion operator is more beneficial for the Sitagliptin MPO task compared to Ranolazine MPO task. The performance improvement in Sitagliptin MPO is much higher compared to the Ranolazine MPO task. This may be because the RNN-LSTM baseline for the Sitagliptin MPO task has more room for improvement, i.e. the generated molecules are, on average, lower scoring. Alternatively, the greedy atom-fusion approach may not be generalisable. That is, parameter searching may be necessary as the optimal atom-level operators' order and their parameters (probability/number of executions) may be different depending on the goal-directed generation task.

Table 7-11. Model Performance on the Sitagliptin MPO Task When Performing Atom-Level Fusion on the Subset Loop Stage of Iterative Fine-Tuning. Results presented are the top-1, top-10, and top-100 MPO scores of the molecules generated. The average column is the mean of the top-1, top-10, and top-100 MPO scores. Each result is the average across three replicates. Brackets indicate the standard deviation. Bold indicates the best augmented result compared to the RNN-LSTM baseline.

	top 1	top 10	top 100	average
RNN-LSTM	0.598 (0.00)	0.530 (0.00)	0.453 (0.00)	0.527 (0.00)
RNN-LSTM Atom-Fusion Subset Loop 1	0.654 (0.00)	0.576 (0.00)	0.485 (0.01)	0.572 (0.00)
RNN-LSTM Atom-Fusion Subset Loop 5	0.796 (0.03)	0.741 (0.04)	0.677 (0.06)	0.738 (0.04)
RNN-LSTM Atom-Fusion Subset Loop 10	**0.807 (0.04)**	**0.789 (0.05)**	**0.730 (0.04)**	**0.775 (0.04)**
Graph GA	0.806 (0.02)	0.790 (0.05)	0.763 (0.05)	0.786 (0.04)

Table 7-12. Model Pserformance on the Ranolazine MPO Task When Performing Atom-Level Fusion on the Subset Loop Stage of Iterative Fine-Tuning. Results presented are the top-1, top-10, and top-100 MPO scores of the molecules generated. The average column is the mean of the top-1, top-10, and top-100 MPO scores. Each result is the average across three replicates. Brackets indicate the standard deviation. Bold indicates the best augmented result compared to the RNN-LSTM baseline.

	top 1	top 10	top 100	average
RNN-LSTM	0.847 (0.01)	0.835 (0.01)	0.814 (0.01)	0.832 (0.01)
RNN-LSTM Atom-Fusion Subset Loop 1	0.844 (0.01)	0.836 (0.01)	0.822 (0.01)	0.834 (0.01)
RNN-LSTM Atom-Fusion Subset Loop 5	0.865 (0.02)	0.857 (0.02)	0.847 (0.02)	0.856 (0.02)
RNN-LSTM Atom-Fusion Subset Loop 10	**0.870 (0.01)**	**0.864 (0.00)**	**0.857 (0.01)**	**0.864 (0.01)**
Graph GA	0.899 (0.01)	0.896 (0.01)	0.888 (0.01)	0.894 (0.01)

7.3.5 Atom-Fusion Operator Investigation without an RNN-LSTM

The results from the previous section indicate that the initial implementation of the greedy atom-fusion with a fixed order of atom-level operators, i.e. atom-swap, atom-delete, and atom-insert, may not be necessarily optimal in the domain of SMILES modification. Therefore, this section investigates how the order of the atom-level operators affects the SMILES augmentation process at the subset loop stage. Additionally, an investigation of the potential additive effects of combining atom-level operators was investigated. Whereby the null hypothesis is that there is no difference in the SMILES scores generated between an atom-fusion operator that uses a combination of three atom-level operators over one that use a combination of only two atom-level operators.

The performance of the atom-fusion operator at the subset loop stage was investigated without an RNN-LSTM model. The aim here is to focus on how the operator is modifying the SMILES without influence of the RNN-LSTM model (Figure 7-15). As a result, the input SMILES here is from a pre-selected dataset rather than the SMILES generated by an RNN-LSTM.

To obtain the initial sample population, the top-1028 SMILES were used to populate the initial results archive at loop 0. The top-512 from the results archived are then taken at each epoch and the fusion operators were applied. The augmented SMILES are then scored and then added to the results archive. For all operator permutations, the loop was performed a total of 500 times (Figure 7-15). At each loop, the MPO score and the contribution score of the top-1 SMILES from the results archive is measured. Note that the top-1 SMILES from the results archive may be the same for a few loops until a higher scoring SMILES is generated. Triplicate repeats were performed for all the permutations and the results reported are the average.

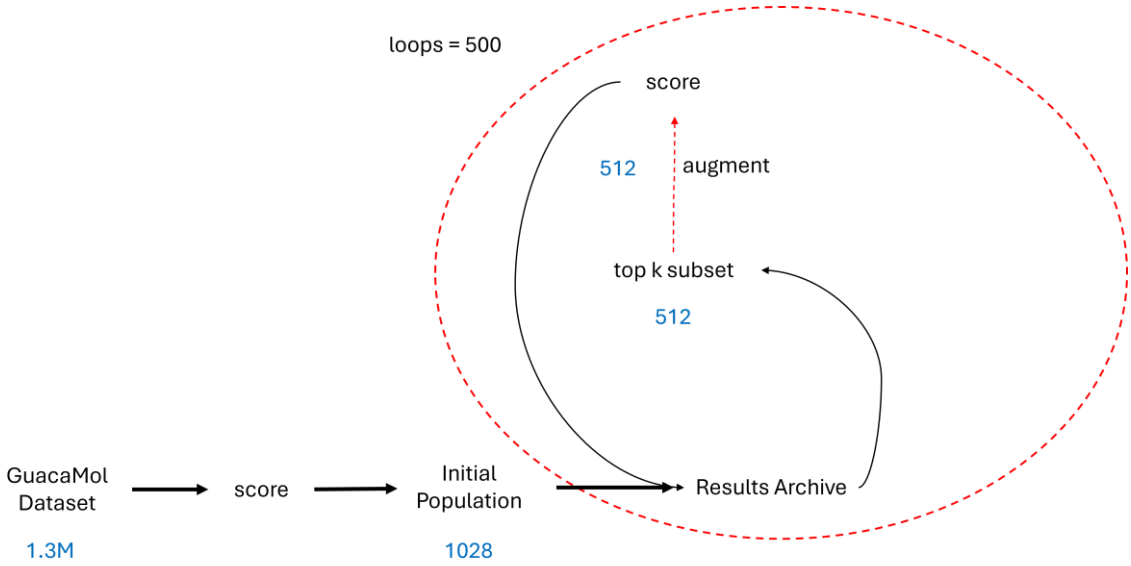


Figure 7-15. External Investigation of the Augmentation-operators. The initial SMILES population is the top-1028 scoring molecules from GuacaMol. The top-512 from the results archive undergo processing with the augmentation operator at each epoch and are added to the results archive. The values shown in blue are the SMILES at each point of the workflow.

Double Operation

The double operation permutation experiments involve finding the optimal order of applying the atom-level operators (swap, insert, and delete) to SMILES. For example, 'SI' indicates that the swap operator is applied first followed by the insert operator. On the other hand, the 'random' fusion is when the operators are chosen at random.

Figure 7-16 and Table 7-13 shows the average Sitagliptin MPO score and the average individual component score of the top-1 SMILES for the different operators across the repeats. The individual component score are the scores that make up the overall Sitagliptin MPO score and include the similarity, the logP, the TPSA, and the isomer scores. The scores at epoch 0 of the trajectory plots are for the top 1028 scoring molecules that were extracted from the GuacaMol training dataset.

Results indicate that the permutations with the insert operators (SI, DI, ID, and IS) perform better than the operators SD and DS. The MPO component scores indicate that the improved performance is due to the atom-insert operator increasing the isomer score (Table 7-13), for example the mean top-1 isomer score for SD (0.547) and DS (0.400) are lower than the scores for SI (0.606), DI (0.883), ID (0.867), and IS (0.635). The score trajectories also indicate that SD and DS plateau at early epochs. This is likely because there are no new tokens introduced in the SMILES to find potential isomers. Interestingly, the results suggest that the order of operation may have some importance. The order of operations achieved different MPO scores, for example SD (0.678) vs DS (0.663), IS (0.791) vs SI (0.745), and DI (0.851) vs ID (0.856). However, these differences are small.

The random atom-fusion achieved a strong result (0.827) (Table 7-13). This method therefore provides an alternative to the greedy atom-fusion as it can perform well without the need of prior parameter searching, which can be computationally expensive and time-consuming.

On the other hand, Figure 7-17 and Table 7-14 shows the average Ranolazine MPO score and the average individual component score of the top-1 SMILES for the different operators across the repeats. The individual component scores are the scores that make up the overall Ranolazine MPO score and includes the similarity, the logP, the TPSA, and the F atom count scores. Results indicate small variations in the trajectory and final top-1 scores. This is likely because it is an easy task to optimise.

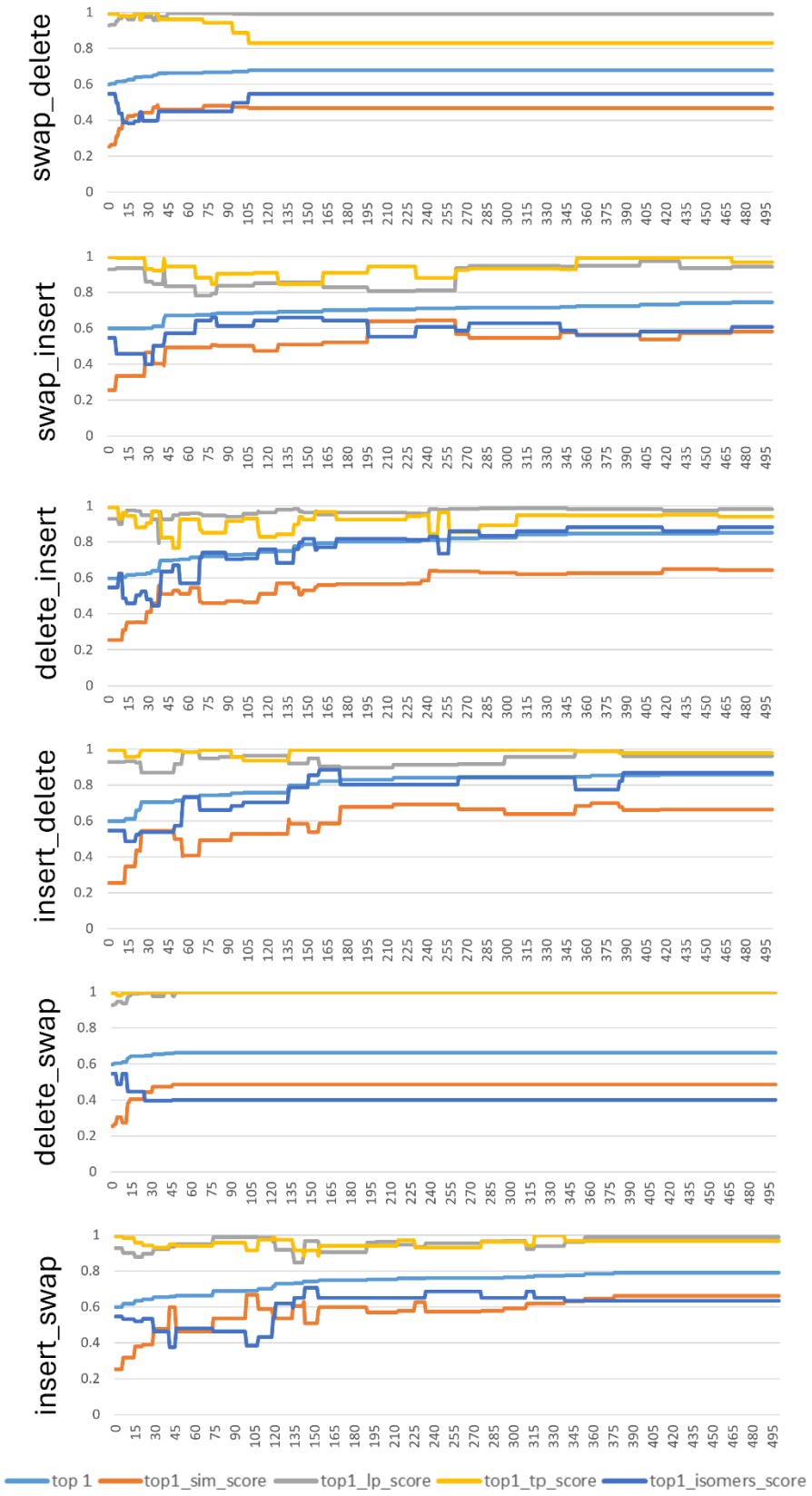


Figure 7-16. The Sitagliptin MPO and its Component Scores Trajectory of the Top-1 SMILES at Each Epoch for the Different Dual Operator Permutations. sim_score – similarity score, lp_score – logP score, tp_score – TPSA score. Each graph is the average scores across three replicates.

Table 7-13. The Top-1 Final Sitagliptin MPO Scores After 500 Epochs for the Double Operator Permutations. Permutations: SD – swap-delete, SI – swap-insert, DI – delete-insert, ID – insert-delete, DS – delete-swap, IS – insert-swap, rand – random fusion with n = 2. . Components: sim_score – similarity score, lp_score – logP score, tp_score – TPSA score. Each result is the average scores across three replicates.

operators	Mean top 1 MPO Score	Mean top 1 Sim_score	Mean top 1 LP_score	Mean top 1 TP_score	Mean top 1 Isomer_score
SD Greedy	0.678	0.469	0.991	0.830	0.547
SI Greedy	0.745	0.583	0.941	0.967	0.606
DI Greedy	0.851	0.643	0.982	0.941	**0.883**
ID Greedy	**0.856**	**0.663**	0.961	0.978	0.867
DS Greedy	0.663	0.485	**0.998**	**0.999**	0.400
IS Greedy	0.791	0.661	0.989	0.967	0.635
Rand - 2	0.827	0.598	0.991	0.980	0.814
model	**Mean top 1 MPO Score**	**Mean top 1 Sim_score**	**Mean top 1 LP_score**	**Mean top 1 TP_score**	**Mean top 1 Isomer_score**
RNN-LSTM	0.598	0.256	0.928	0.993	0.547
Graph-GA	0.806	0.517	0.951	0.968	0.892

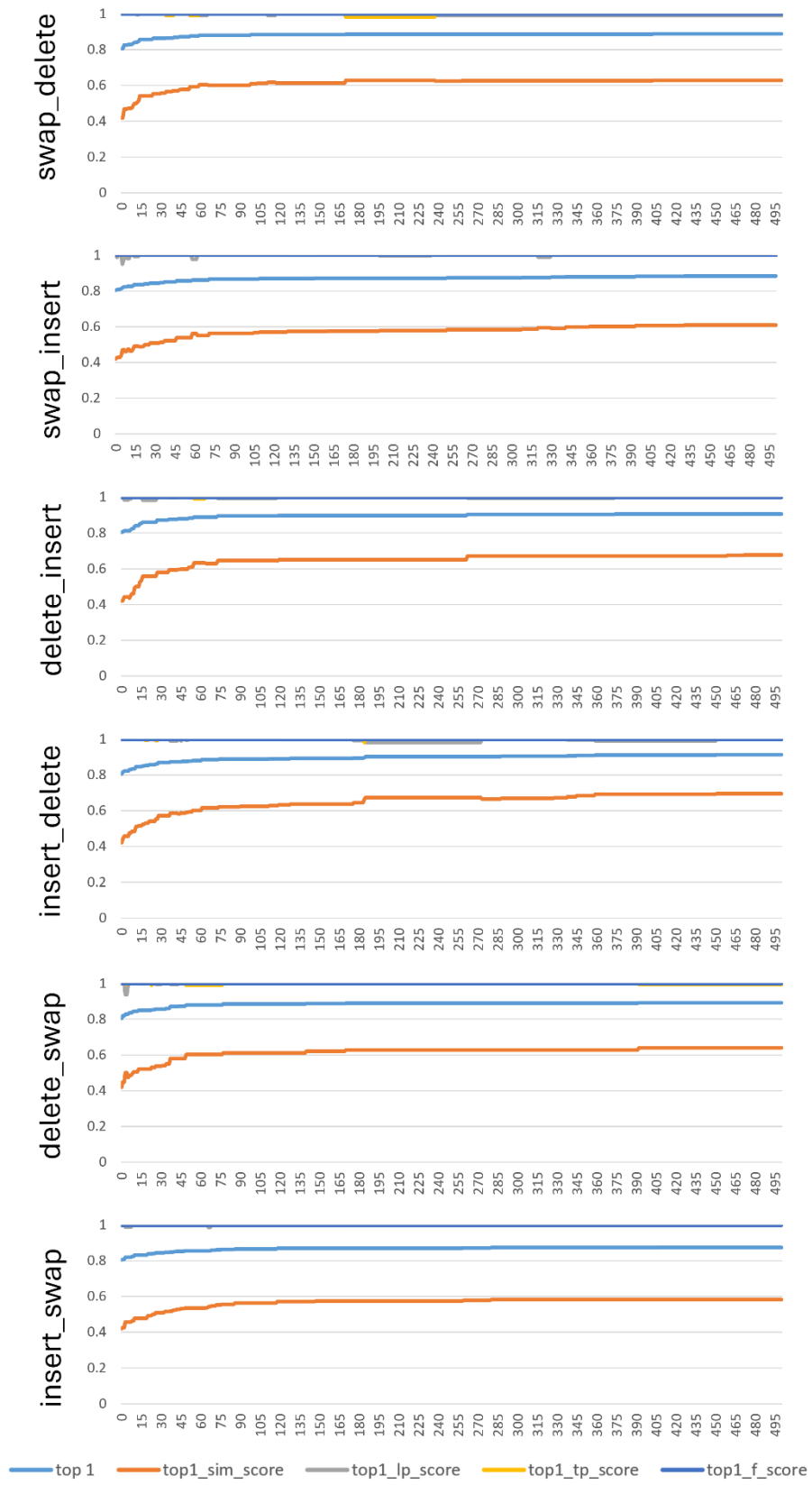


Figure 7-17. The Ranolazine MPO and its Component Scores Trajectory of the Top-1 SMILES at Each Epoch for the Different Dual Operator Permutations. components: sim_score – similarity score, lp_score – logP score, tp_score – TPSA score, f_score – F atom count score. Each graph is the average scores across three replicates.

Table 7-14. The Top-1 Final Ranolazine MPO Scores After 500 Epochs for the Different Dual Operator Permutations. Permutations: SD – swap-delete, SI – swap-insert, DI – delete-insert, ID – insert-delete, DS – delete-swap, IS – insert-swap, rand – random fusion with n = 2. Components: sim_score – similarity score, lp_score – logP score, tp_score – TPSA score, f_score – F atom count score. Each result is the average scores across three replicates.

operators	Mean top 1 MPO Score	Mean top 1 Sim_score	Mean top 1 LP_score	Mean top 1 TP_score	Mean top 1 F_score
SD Greedy	0.888	0.628	0.991	**1.000**	**1.000**
SI Greedy	0.884	0.610	**1.000**	**1.000**	**1.000**
DI Greedy	0.907	0.677	0.999	**1.000**	**1.000**
ID Greedy	**0.913**	**0.695**	**1.000**	**1.000**	**1.000**
DS Greedy	0.892	0.640	0.995	0.994	**1.000**
IS Greedy	0.874	0.582	**1.000**	**1.000**	**1.000**
Rand - 2	0.895	0.653	**1.000**	0.983	**1.000**
model	**Mean top 1 MPO Score**	**Mean top 1 Sim_score**	**Mean top 1 LP_score**	**Mean top 1 TP_score**	**Mean top 1 F_score**
RNN-LSTM	0.851	0.525	1.000	1.000	1.000
Graph-GA	0.890	0.626	1.000	1.000	1.000

Triple Operation

The triple operation permutation experiments involve finding the optimal order of applying the atom-level operators (swap, insert, and delete). For example, 'SDI' indicates that the swap, delete, insert operators are applied in the given order. On the other hand, the 'random' fusion is when three operators are chosen at random.

Figure 7-18 and Table 7-15 shows the average Sitagliptin MPO score and the average individual component scores of the top-1 SMILES for the different triple atom-fusion operators across the repeats. Results from the triple operation experiments also indicate that there may be some importance to the order of operations as different permutations achieve some differences in their results. The result with a clear discrepancy is the top-1 similarity score for ISD (0.618) and IDS (0.720) (Figure 7-18 and Table 7-15). Additionally, the SID, ISD, DSI, and IDS permutations seem to be more stable at later epochs (Figure 7-18). Finally, the random atom-fusion scored the lowest in the MPO score (0.822).

Figure 7-19 and Table 7-16 shows the average Ranolazine MPO score and the average individual component scores of the top-1 SMILES for the different triple atom-fusion operators across the repeats. Similar to the double operator results, only small variations in the trajectory and final top-1 scores are observed. This is likely because it is an easy task to optimise.

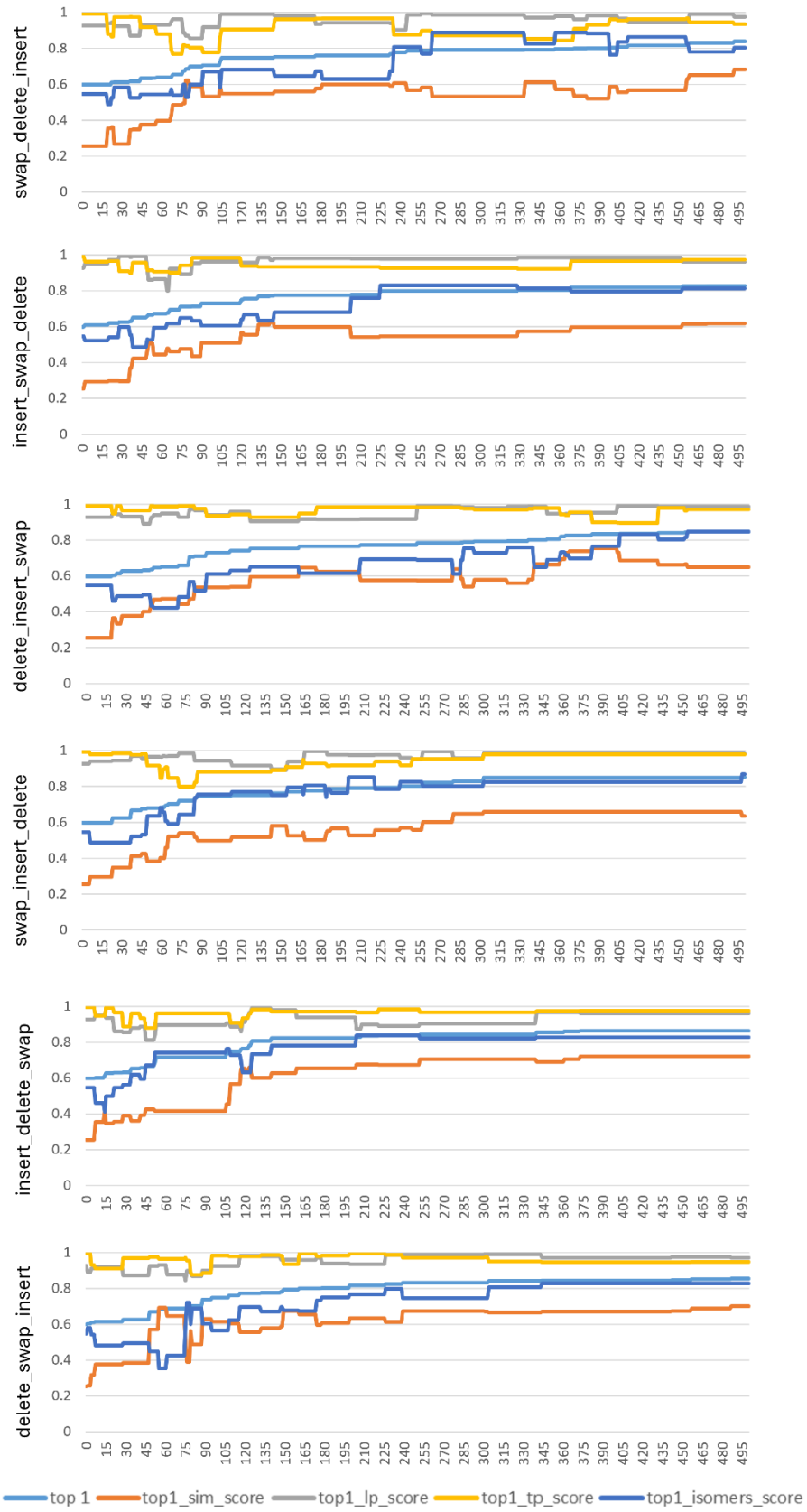


Figure 7-18. The Sitagliptin MPO and its Component Scores Trajectory of the Top-1 SMILES at Each Epoch for the Different Triple Operator Permutations. Components: sim_score – similarity score, lp_score – logP score, tp_score – TPSA score. Each graph is the average scores across three replicates.

Table 7-15. The-Top 1 Final Sitagliptin MPO Scores After 500 Epochs for the Triple Operator Permutations. Permutations: SDI – swap-delete-insert, ISD – insert-swap-delete, DIS – delete-insert-swap, SID – swap-insert-delete, IDS – insert-delete-swap, DSI – delete-swap-insert, rand – random fusion with n = 3. Components: sim_score – similarity score, lp_score – logP score, tp_score – TPSA score. Each result is the average scores across three replicates.

operators	Mean top 1 MPO Score	Mean top 1 Sim_score	Mean top 1 Lp_score	Mean top 1 Tp_score	Mean top 1 Isomer_score
SDI Greedy	0.840	0.684	0.977	0.935	0.803
ISD Greedy	0.827	0.618	0.961	0.973	0.813
DIS Greedy	0.847	0.650	**0.986**	0.972	0.847
SID Greedy	0.852	0.635	0.982	**0.979**	**0.868**
IDS Greedy	**0.863**	**0.720**	0.960	0.975	0.828
DSI Greedy	0.855	0.701	0.971	0.949	0.829
Rand - 3	0.822	0.608	0.982	0.968	0.791
model	**Mean top 1 MPO Score**	**Mean top 1 Sim_score**	**Mean top 1 LP_score**	**Mean top 1 TP_score**	**Mean top 1 Isomer_score**
RNN-LSTM	0.598	0.256	0.928	0.993	0.547
Graph-GA	0.806	0.517	0.951	0.968	0.892

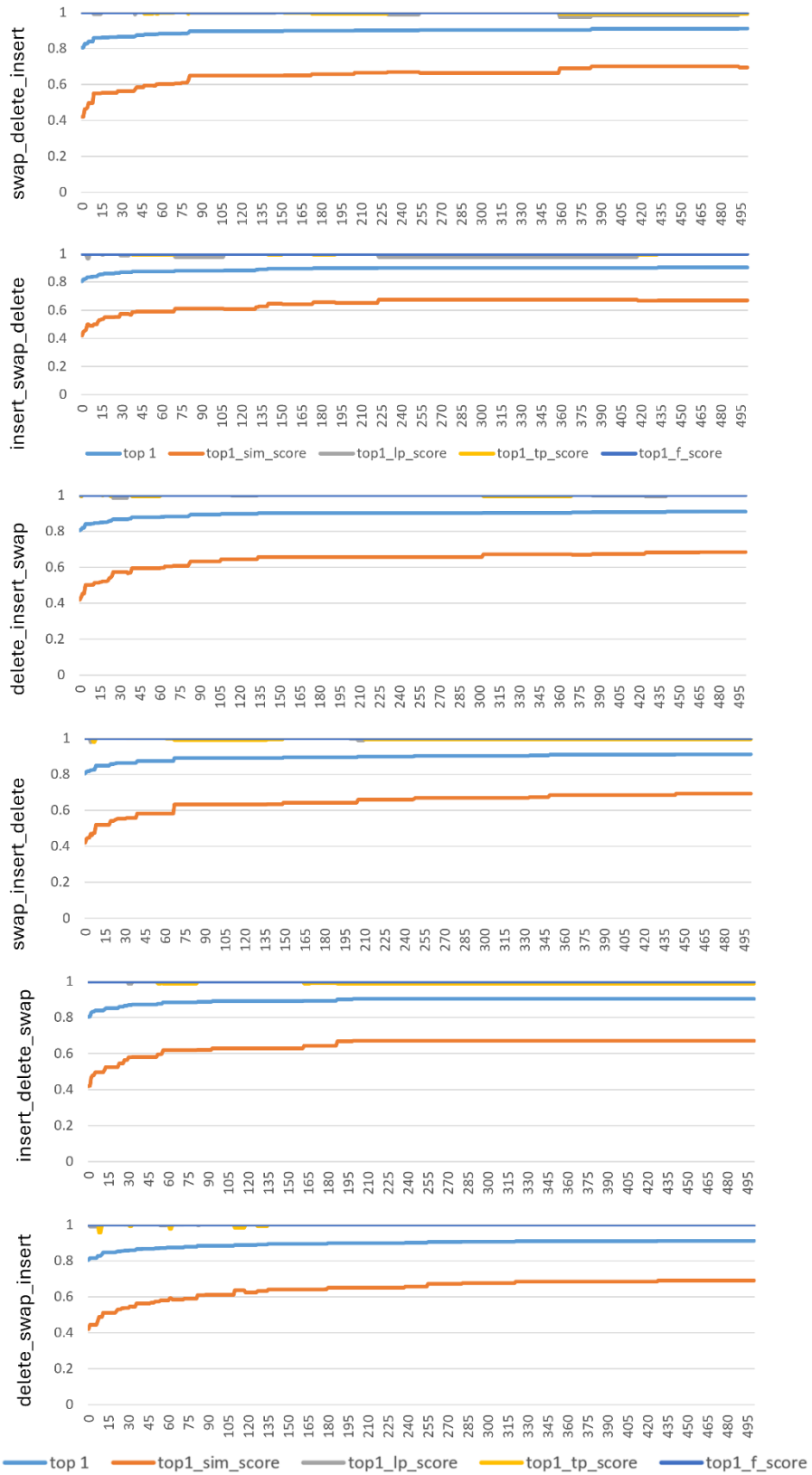


Figure 7-19. The Ranolazine MPO and its Component Scores Trajectory of the Top-1 SMILES at Each Epoch for the Different Triple Operator Permutations. components: sim_score – similarity score, lp_score – logP score, tp_score – TPSA score, f_score – F atom count score. Each graph is the average scores across three replicates.

Table 7-16. The Top-1 Final Ranolazine MPO Scores After 500 Epochs for the Different Dual Operator Permutations. Permutations: SDI – swap-delete-insert, ISD – insert-swap-delete, DIS – delete-insert-swap, SID – swap-insert-delete, IDS – insert-delete-swap, DSI – delete-swap-insert, rand – random fusion with n = 3. Components: sim_score – similarity score, lp_score – logP score, tp_score – TPSA score, f_score – F atom count score. Each result is the average scores across three replicates.

operators	Mean top 1 MPO Score	Mean top 1 Sim_score	Mean top 1 LP_score	Mean top 1 TP_score	Mean top 1 F_score
SDI Greedy	0.912	0.695	1.000	0.994	1.000
ISD Greedy	0.905	0.670	1.000	1.000	1.000
DIS Greedy	0.909	0.684	1.000	1.000	1.000
SID Greedy	0.911	0.694	1.000	0.994	1.000
IDS Greedy	0.903	0.672	1.000	0.989	1.000
DSI Greedy	0.912	0.691	1.000	1.000	1.000
Rand - 3	0.880	0.604	0.994	1.000	1.000
model	**Mean top 1 MPO Score**	**Mean top 1 Sim_score**	**Mean top 1 LP_score**	**Mean top 1 TP_score**	**Mean top 1 F_score**
RNN-LSTM	0.851	0.525	1.000	1.000	1.000
Graph-GA	0.890	0.626	1.000	1.000	1.000

Discussion

The results of the operator permutation experiments suggest some importance in operator ordering as the different permutations achieved different results in the MPO scores and its components. Additionally, the operators do not seem to be additive as some of the double operators outperformed the triple operators, for example the MPO scores of the ID (0.856) vs SDI (0.840), ISD (0.827), DIS (0.847), SID (0.852), DSI (0.855) and Rand-3 (0.822). Finally, the random fusions do not perform considerably worse than the fixed order atom-fusion operators and will therefore be used for subsequent experiments using atom-fusion operators.

7.4 Conclusion

Deep learning-based methods require large datasets to perform well. A common way to address this is the use of data augmentation to artificially expand the training dataset. The present work proposed NLP-inspired operators for data augmentation in the deep *de novo* design domain.

The results from this study indicate that data augmentation using NLP-inspired operators improved model performance when compared to the baseline. Combining these operators (i.e. the fusion method) improved model performance much more than the individual operators. However, a clear limitation of the method proposed here is that it can be computationally inefficient as it increases the number of scoring function calls. On average, the baseline RNN-LSTM performs, a total of 10,000 scoring function calls. On the other hand, a total average of 40,000 scoring function calls is performed with the inclusion of an augmentation operator at the subset loop with l = 5. This high number of scoring function calls can be problematic for more expensive scoring functions. Additionally, the operators currently do not incorporate chemical knowledge. This may have a negative impact on the synthetic accessibility of the generated molecules and/or drug-likeness. Therefore, the next chapter evaluates the synthetic accessibility and drug-likeness of the molecules generated using the NLP operators and then introduces some new operators that incorporate chemical knowledge with the aim of generating molecules that are more likely to be synthetically accessible.

Chapter 8. Chemically-sensible Operators

8.1 Introduction

The main limitation of generative *de novo* design models is that they propose molecules that are hard to synthesise. Synthesisability is an important consideration in the drug design process because although models may be able to propose highly optimised molecules, they need to be synthesised to be validated and used. Therefore, there is an increasing focus on considering the synthesisability of the molecules proposed by generative *de novo* design models (Stanley and Segler, 2023).

Although the set of atom-level operators described in the previous chapter were successful in improving the optimisation score of the generated molecules, they achieved this while sacrificing on synthesisability. Multiple methods have been proposed to improve the synthesisability of deep *de novo* design models, these include dataset bias, post-hoc filtering, synthesisability-constrained generation, and integration of synthesisability into scoring functions, as discussed in the literature review chapter (Gao and Coley, 2020). Post-hoc filtering, in particular, has the advantage of not being as resource-intensive compared to the other methods.

This chapter first quantifies the synthesisability of molecules generated with and without the data augmentation operators. Then, novel methods not previously explored in the literature are introduced which aim to improve the synthesisability of generated molecules while also improving the optimisation score. The first method is to impose constraints on the atom-level operators by masking substructures that may be important for the synthetic accessibility of the molecule. The next set of methods represent more chemically meaningful data augmentation operators. These are based on operators in the wider literature and have not previously been applied to data augmentation for *de novo* design and/or goal-directed generation. They include graph-based operators and bioisosteric substitution. The final section investigates post-hoc filtering as a potential method of improving the synthesisability of the generated molecules, called finetuning dataset filtering in this work.

8.2 Methods

8.2.1 Quantifying Chemical Sensibility

As seen in the previous chapter, visual inspection of the generated molecules from the augmented RNN-LSTMs suggests that the proposed operators have a negative impact on the 'chemically-sensibility' of the outputs. In this chapter, synthetic accessibility score (SAscore) and quantitative estimate of drug-likeness (QED) are used to quantify the 'chemical-sensibility' of the generated molecules.

SAscore quantifies how easily synthesisable a given molecule is by summing the fragment score and the structural complexity score (Ertl and Schuffenhauer, 2009). The implementation used in this study is from RDKit and also considers a molecule's symmetry (equation (*9*). A high fragment score and symmetry indicates that a molecule is more likely to be synthesisable. In contrast, a high structural complexity score indicates that a molecule is less likely to be synthesisable. The final SAscore scales and transforms equation (*9* such that values range from 1 (easy to synthesise) to 10 (difficult to synthesise).

$$Penalities = fragment\ score - structural\ complexity + symmetry\ correction \quad (9)$$

The fragment score is used later on for the SAscore-based masking and is therefore described further here. The score is based on a dictionary of "fragments" where a fragment is an atom and its environment represented as a Morgan fingerprint of radius 2. The dictionary of fragments is based on an analysis of 934,046 PubChem molecules that have been fragmented with each fragment assigned a score that reflects its frequency of occurrence in the set of molecules (Ertl and Schuffenhauer, 2009). The fragment score is in the range between -4 and 3. The assumption is that fragments commonly found in drug-like molecules are easier to synthesise and will be given a higher scores (closer to 3) and fragments that are less common or rare in the database are penalised with lower scores (closer to -4). When SAscore is applied to score a whole molecule, the scores for all fragments are retrieved from the dictionary, summed, and then normalised with the number of total fragments.

QED is a measure of drug-likeness, i.e. it quantifies how well a given molecule fits within the chemical space of known drugs. QED is calculated using multiple properties and structural features, which include molecular weight, logP, numbers of hydrogen bond donors and acceptors, number of aromatic rings, topological surface area, number of rotatable bonds, and presence of unwanted substructures. The final output of the QED score is in the range of 0-1, where a higher QED score indicates that a given molecule is more "drug-like" (Bickerton *et al.*, 2012). The implementation used in this study is from RDKit.

The SAscores and QED scores are complementary indicators for the ideal drug candidate. However, it is important to note there are important considerations when interpreting these values. Synthesisabilty does not indicate biological relevance or pharmacological activity and may penalise novel scaffolds that may be useful but are not well-known in synthetic chemistry. Similarly, many successful drugs violate drug-likeness rules, e.g. natural products.

8.2.2 Fine-tuning Dataset Filtering

Filtering is used to remove molecules that are potentially problematic in drug discovery projects, for example, molecules that may contain potentially unwanted functional groups, substructures or properties. A common filtering workflow is to predefine a list of unwanted features using SMARTS and then remove any molecules from the dataset that match these SMARTS.

This work uses rd_filters' structural alerts list, a fixed version of ChEMBL20's list, available at (https://github.com/PatWalters/rd_filters) (Walters, 2018). The ChEMBL20's list of alerts contains SMARTS patterns that have originally been extracted from multiple publicly available rule sets that were defined independently and used for different applications (listed below). However, several SMARTS from the original ChEMBL20's list are incompatible with RDKit because of issues with syntax. These incompatible SMARTS patterns were modified by the author so that they can be parsed by RDKit.

Pfizer's LINT (LINT) filters contain 57 structural alerts which identify substructures observed to lead to suboptimal physicochemical properties for oral absorption during preclinical optimisation (Blake, 2005). Glaxo Wellcome Hard (Glaxo) filters contain 55 structural alerts that identify functional groups that are correlated with reactivity in high-throughput screening (Hann *et al.*, 1999). Bristol-Myers Squibb HTS Deck (BMS) filters contain 180 structural alerts that identify undesirable compounds that typically contributes to promiscuity in high-through screening (Pearce *et al.*, 2006). NIH MLSMR Excluded Functionality (MLSMR) filters contain 116 structural alerts that exclude compounds with functional groups that would interfere in HTS and compounds likely to be promiscuous aggregators (NIH Molecular Libraries, 2013). University of Dundee NTD Screening Library (Dundee) filters contains 105 structural alerts that identify functional groups that are mutagenic (e.g. nitro groups), have unfavourable properties (e.g. sulfates and phosphates), and reactive functional groups (e.g. 2-halopyridines and thiols) (Brenk *et al.*, 2008). PAINS filters contain 481 structural alerts that identify substructural features that appear as frequent promiscuous hitters in high-throughput screening and lead to false positive results (Baell and Holloway, 2010). SureChEMBL Non MedChem-Friendly SMARTS (SureChEMBL) filters contain 166 structural alerts that identify substructures that are highly correlated with undesirable properties. Note, some of the collections include the same SMARTS patterns. For example, the SMARTS *=C=* is present in both SureChEMBL and MLSMR.

Filtering was incorporated in the optimisation process by applying one of the rd_filters rule sets on the augmented SMILES output during the augmentation process (Figure 8-1). The molecules remaining

after undesirable molecules were removed were the filtered molecules. The filtered molecules are then scored and added to the results archive. The results archive was therefore updated to have more samples that are drug-like, and the model can therefore be biased towards generating more drug-like molecules. The RNN-LSTM baseline model was also filtered for comparison. Here, the top-k subset obtained from the results archive, without data augmentation, was filtered before being used to fine-tune the RNN-LSTM model.

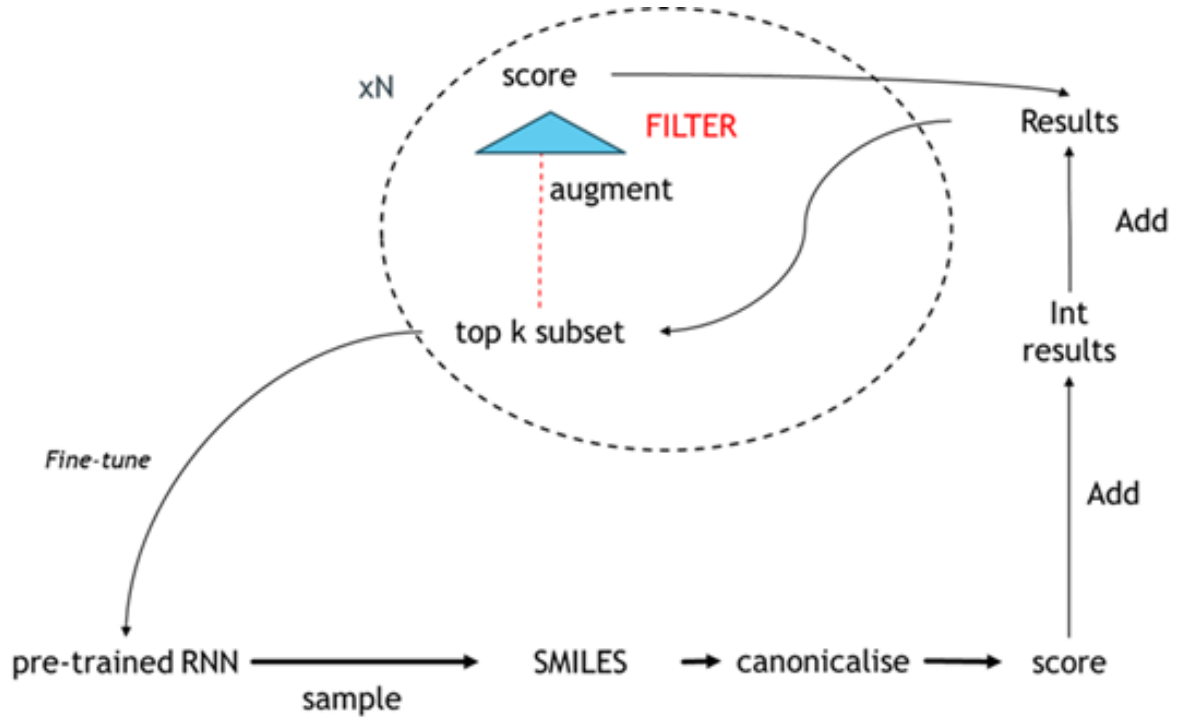


Figure 8-1. Dataset Filtering during Fine Tuning via Subset Loop Augmentation. The figure shows that after the augmentation operation the augmented molecules are filtered before being scored and stored in the results archive.

8.2.3 Substructure Masking

The substructure masking approach aims to preserve important features that have been identified as being common in bioactive molecules by masking them during the data augmentation process.

SAscore-based Atom Masking

The aim here is to preserve atoms that are predicted to be in a synthesisable environment which should limit the loss in synthesisability. The method is referred to as SAscore-based atom masking and involves firstly scoring the atoms in a molecule using the fragment score component of the SAscore method, discussed above, and preventing augmentation operators from editing atoms based on the scores.

For the SAscore-based atom masking methods, scores are associated with each atom in a molecule rather than being summed over the whole molecule. This mapping is required as the NLP-operators work at the atom-level. The first step is to generate sparse count Morgan fingerprints for a molecule. The score for each fingerprint is then retrieved from the precomputed dictionary, in turn. Each score is mapped from the fragment-level to the atoms found in the fragment (both the central atom and the neighbouring atoms within a radius of 2) as described below. Like the fragment-level score, the atom-level SAscore is in the range between -4 and 3. Atoms that likely to be in a synthesisable environment is given a score closer to 3, and vice versa. A threshold can then be defined where atoms that have a higher value will be masked, called SAS mask. For example, if an SAS mask with a threshold of 2 is set, only atoms with lower atom-level SAscore will be edited by the operators.

The mapping method described below is inspired by a previous work where fingerprint bit-level attributions were mapped to atom-level attributions to indicate atoms that contribute to toxicity prediction (Walter, Webb and Gillet, 2024). The steps are as follows:

1. Generate Morgan fingerprints of radius 2 for the molecule.

   RDKit's GetSparseCountFingerprint() function is used to generate fingerprints to represent molecular fragments. This is RDKit's implementation of the Extended Connectivity Fingerprint (ECFP), i.e. a circular environment around an atom up to a radius 2. The fingerprints can then be used as a fragment identifier for the subsequent step.

2. Retrieve the fragment ID scores, i.e. the fragment-level scores.

   The next step is to perform a lookup of the scores of the fragment IDs from the precomputed fragment score dictionary (Step 2 in Figure 8-2). The fragment scores ranges from -4 to 3. If the fragment ID is not present in the dictionary, the fragment is automatically given a score -4. This means that novel or unusual fragments are considered difficult to synthesise.

3. Aggregating Atom Scores

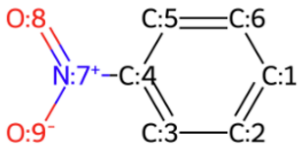
   For each fragment ID in the molecule, the fragment-level score is appended to the central atom and its neighbouring atoms (up to a radius of 2). After iterating through all of the fragment IDs, each atoms in the molecule will have a list of scores obtained from different fragments. Atoms may have more than one score as some atoms may be encoded as a neighbour in another central atom. For example, in Figure 8-2, fragment 951226070 has a fragment-level score of 2.79. The atoms 3 (C) and 5 (C) are encoded in this fragment. Therefore, the fragment-level score 2.79 is appended to both atoms.

4. Averaged Atom Score

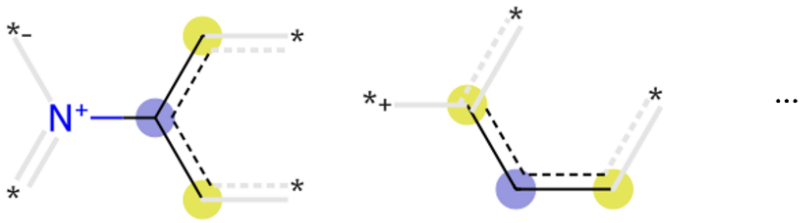
   Finally, the final atom-level score is calculated as the average of each atom's list of appended scores. For example, atom 7 (N) has the appended scores of 1.53, 1.55, and 1.24. This is averaged to give the final atom score of 1.44. These atoms-level scores can then be used to mask atoms based on a user-defined threshold.

SMILES/RDKit Mol Object

c1ccc(cc1)[N+](=O)[O-]



1. Morgan Fingerprint (Fragments)



2. Fragment Score

Fragment ID: 1428903162	Fragment ID: 951226070
Score: 1.24	Score: 2.79
Atoms: 4	Atoms: 3, 5

3. Aggregated Atom Score

1: [2.5981, 2.1353, 2.8295],
2: [2.5981, 2.8295, 2.4424],
3: [2.7908, 0.5843, 2.8295],
4: [1.238, 0.7921, 2.8416],
5: [2.7908, 0.5843, 2.8295],
6: [2.5981, 2.8295, 2.4424],
7: [1.5281, 1.5507, 1.2364],
8: [2.7721, 1.5463],
9:[1.557, 1.5476]

4. Averaged Atom Score

1: 2.5201,
2: 2.6233,
3: 2.0682,
4: 1.6239,
5: 2.0682,
6: 2.6233,
7: 1.4384,
8: 2.1592,
9: 1.552

Figure 8-2. Atom-Level SAscore of Nitrobenzene. 1, The first step involves generating sparse count Morgan fingerprints for the molecule to represent the fragments **2,** The scores of the fingerprints are then identified from a precomputed dictionary from RDkit's SAscore component. **3,** For each fingerprint, the score is mapped to the central atom and its neighbouring atoms (up to a radius of 2). Atoms may inherit more than one score because they may be encoded as a neighbour in other central atoms. **4,** For each atom, the appended scores are averaged to give the final atom-level SAscore.

As a proof of concept, Figure 8-3 shows an example where there are sections of the molecule that can be considered not to be chemically sensible. The cumulene substructure, i.e. the chain of double bonds from atoms 1-6, has correctly been identified to consists of atoms in an unfavourable environment, as shown by the negative atom scores. Cumulene is known to be hard to synthesise due to their instability (Kim, 2009). Secondly, the four-membered ring (atoms 7, 8, 14, and 27) has been correctly identified to be atoms in an unfavourable environment as these rings are typically unstable due to ring strain. Also, the insertion of a tetrazole into this four-membered ring may be difficult due to the ring strain and may explain the negative value of atom 9 (N). Finally, the highly fluorinated five-membered ring (atoms 16, 19, 21, 23, 25) is also unfavourable. This is likely because introducing multiple fluorenes selectively is synthetically challenging due to their reactivity (Ishida, Sheppard and Nishikata, 2018). Additionally, this may make it challenging to introduce sulfur into the highly fluorinated ring and may explain the negative score of atom 17 (S).

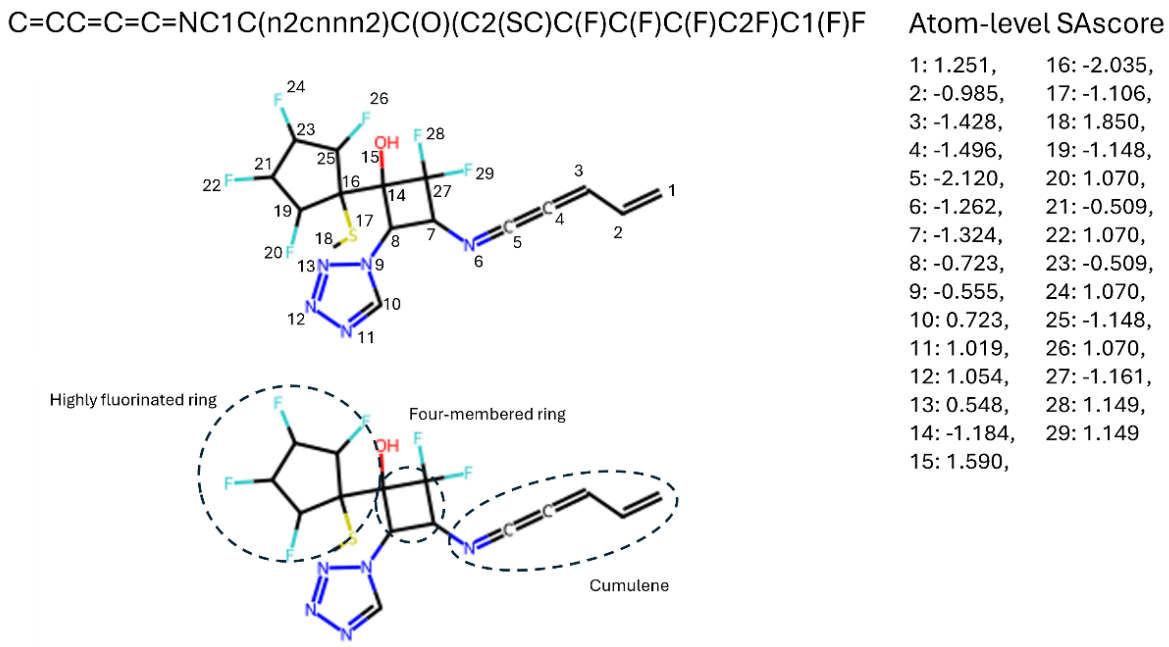


Figure 8-3. Atom-Level SAscore on a Molecule Containing Undesirable Substructures. The figure shows how the atom-level SAscoring identifies the unwanted substructures within a molecule. The SMILES' atoms were numbered and scored as shown on the right.

In contrast, Figure 8-4 shows an example where atom-level SAscoring is applied to a molecule containing functional groups known to be desirable in medicinal chemistry, which includes chlorine, carboxamide, and phenol groups. These groups are commonly found in small-molecule drugs and are often synthesised, and are considered synthesisable (Scott, Cox and Njardarson, 2022; Joshi and Srivastava, 2023; Yang *et al.*, 2025). These examples suggests that the atom-level SAscore implementation is able to quantify both unreasonable and reasonable sections of the SMILES.

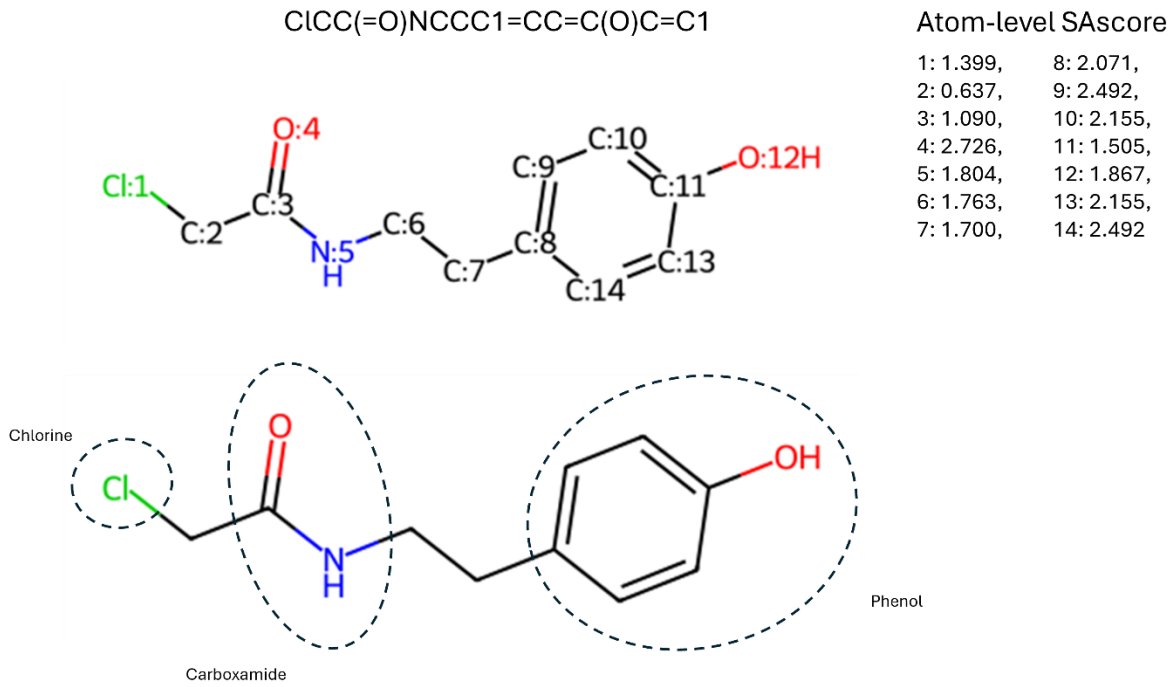


Figure 8-4. Atom-Level SAscores of a Molecule Containing Desirable Substructures. The figure shows how the atom-level SAscoring identifies the desirable substructures within a molecule. The SMILES' atoms were numbered and scored as shown on the right.

Functional Group Masking

Masking functional groups that are ‘chemically-relevant’ is another alternative to mitigate loss of synthetic accessibility during data augmentation.

Figure 8-5 illustrates the workflow of the method. First a SMILES, is encoded into an encoded SMILES (eSMILES) and an RDKit mol object. As discussed in the previous chapter, encoding SMILES allow the atom-level operators to handle double-character elements, e,g. Cl is encoded as an ‘X’ in Figure 8-5, whereas the RDKit mol object allows the use of SMARTS matching. The next step is to identify functional groups of interest in the molecule. Here, the SMARTS pattern of 50 functional groups that are commonly found in bioactive molecules are used (Ertl, Altmann and McKenna, 2020). Atoms matching any of the SMARTS functional groups are mapped to eSMILES indices. Finally, the eSMILES index of the functional groups of interest are masked, preventing editing operations.

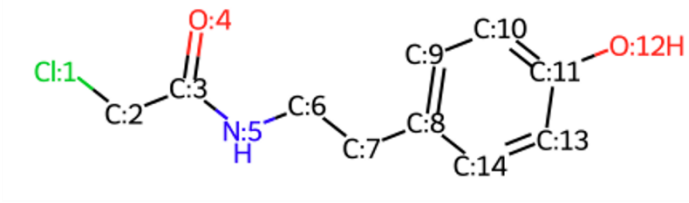
SMILES:

ClCC(=O)NCCC1=CC=C(O)C=C1

Encoded SMILES:

XCC(=O)NCCC1=CC=C(O)C=C1

Mol Object:



Atom Object Index:

[(1, 'Cl'), (2, 'C'), (3, 'C'), (4, 'O'), (5, 'N'), (6, 'C'), (7, 'C'),
(8, 'C'), (9, 'C'), (10, 'C'), (11, 'C'), (12, 'O'), (13, 'C'), (14, 'C')]

SMARTS Matching

SMARTS Matches:

{'*N*': [[3, 5, 6]], 'Cl*': [[1, 2]], 'O*': [[12, 11]],
'*C(*)=O': [[2, 3, 5, 4]], 'N*': [[5, 3], [5, 6]]}

Atom Obj idx -> eSMILES idx Mapping

Masked Encoded SMILES Index

{0, 1, 2, 5, 7, 8, 16, 18}

X	C	C	(	=	O	)	N	C	C	C	1	=	C	C	=	C	(	O	)	C	=	C	1
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23

Figure 8-5. Functional Group Masking. The first step involves converting the SMILES into encoded SMILES (eSMILES) and RDKit mol object. The RDKit mol object is used to find functional groups that match SMARTS patterns of interest. These matches are in the form of atom object index and are mapped to the corresponding eSMILES index. Finally, the identified eSMILES idx are masked from editing operations (as shown by the greyed out indexes). The atom-level operators work on the eSMILES to allow the handling of two-character elements.

8.2.4 Chemistry-based Data Augmentation

Structural transformation is an essential tool in drug design to optimise drug candidates. Examples include SMARTS reactions (Daylight Chemical Information Systems, no date) and medicinal chemistry transformations (bioisosteric substitution) (Patani and LaVoie, 1996). These transformations are more likely to produce chemically sensible molecules and are therefore explored as different forms of data augmentation operators.

8.2.4.1 Graph-based Operations

The use of graph-based representations allows for manipulations that are more likely to be chemically sensible. Whereas the previous operators were based on NLP operators and treated the molecules as text strings, the graph-based operators consider the graph representations of molecules directly where the atoms and nodes are considered as structural elements. The graph mutation operators from Jensen (2019) are used as data augmentation operators. Examples of augmented SMILES are shown in Figure 8-6.

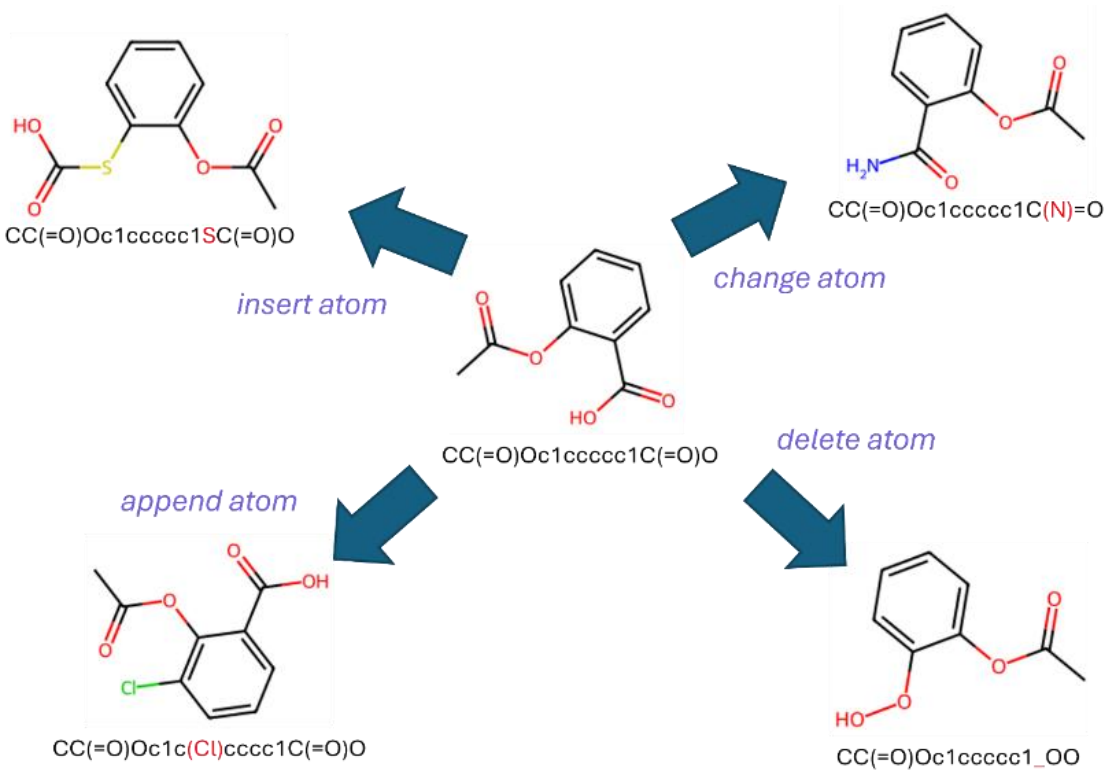


Figure 8-6. Examples of Graph-based Operations applied to Aspirin.

Graph-Fusion Operator

The results from the previous chapter indicate that combining the different atom-level operators (atom-delete, atom-insert, and atom-swap) into the atom-fusion operator improved the MPO score the most. Therefore, the graph-based operators from Jensen (2019) has been combined into the graph-fusion operator. The workflow of the graph-fusion operator is described below:

1. Convert SMILES into an RDKit mol object.
2. Choose a random operator. A list of the mutation operators includes:
    - Append atom – add a new atom by appending it as a side chain
    - Insert atom – inserts a new atom into an existing bond
    - Delete atom – deletes an atom based on the valence and surrounding structure
    - Change atom type – Changes an atom to a different atom
    - Change bond order – Changes a bond to a different bond
    - Delete Ring bond – Deletes a bond within a ring
    - Add ring bond – Forms a ring from a chain of atoms
3. For the chosen operator, choose a mutation subtype, defined by a SMARTS pattern, based on a probability.

    The SMARTS patterns were made by Jensen (2019) to be similar to the mutation operations found in the Algorithm for Chemical Space Exploration with Stochastic Search (ACSESS) algorithm (Virshup *et al.*, 2013). See Table 8-1 for more details.
4. Perform a SMARTS reaction on the mol object using the chosen operator.
5. Repeat steps 2-4 n-times.
6. The final augmented mol object is then converted back into SMILES string.

Table 8-1. Graph Operator's List of Mutations and SMARTS Patterns. The probability is the chance of selecting a specific mutation's SMARTS pattern.

Mutation	SMARTS Pattern	Probability
Append Atom	[*;!H0:1]>>[*:1]X X= 'C', 'N', 'O', 'F', 'S', 'Cl', 'Br'	0.60
	[*;!H0;!H1:1]>>[*:1]X X = 'C', 'N', 'O'	0.35
	[*;H3:1]>>[*:1]X X = 'C', 'N'	0.05
Insert Atom	[*:1]~[*:2]>>[*:1]X[*:2] X = 'C', 'N', 'O', 'S'	0.60
	[*;!H0:1]~[*:2]>>[*:1]=X-[*:2] X = 'C', 'N'	0.35
	[*;!R;!H1;!H0:1]~[*:2]>>[*:1]#X-[*:2] X = 'C'	0.05
Delete Atom	[*:1]~[D1:2]>>[*:1]	0.25
	[*:1]~[D2:2]~[*:3]>>[*:1]-[*:3]	0.25
	[*:1]~[D3:2](~[*;!H0:3])~[*:4]>>[*:1]-[*:3]-[*:4]	0.25
	[*:1]~[D4:2](~[*;!H0:3])(~[*;!H0:4])~[*:5]>>[*:1]-[*:3]-[*:4]-[*:5]	0.19
	[*:1]~[D4:2](~[*;!H0;!H1:3])(~[*:4])~[*:5]>>[*:1]-[*:3](-[*:4])-[*:5]	0.06
Change Atom Type	[X:1]>>[Y:1] X = 'C', 'N', 'O', 'F', 'S', 'Cl', 'Br' Y = 'C', 'N', 'O', 'F', 'S', 'Cl', 'Br'	C = 0.15, N = 0.15, else = 0.14
Change Bond Order	[*:1]!-[*:2]>>[*:1]-[*:2]	0.45
	[*;!H0:1]-[*;!H0:2]>>[*:1]=[*:2]	0.45
	[*:1]#[*:2]>>[*:1]=[*:2]	0.05
	[*;!R;!H1;!H0:1]~[*:2]>>[*:1]#[*:2]	0.05
Delete Ring Bond	[*:1]@[*:2]>>([*:1].[*:2])	NA
Add Ring Bond	[*;!r;!H0:1]~[*;!r:2]~[*;!r;!H0:3]>>[*:1]1~[*:2]~[*:3]1	0.05
	[*;!r;!H0:1]~[*!r:2]~[*!r:3]~[*;!r;!H0:4]>>[*:1]1~[*:2]~[*:3]~[*:4]1	0.05
	[*;!r;!H0:1]~[*!r:2]~[*:3]~[*:4]~[*;!r;!H0:5]>>[*:1]1~[*:2]~[*:3]~[*:4]~[*:5]1	0.45
	[*;!r;!H0:1]~[*!r:2]~[*:3]~[*:4]~[*!r:5]~[*;!r;!H0:6]>>[*:1]1~[*:2]~[*:3]~[*:4]~[*:5]~[*:6]1	0.45

8.2.4.2 Medicinal Chemistry Transformation

In medicinal chemistry, there is a set of structural transformations that has been observed to be historically beneficial in rational drug design. Examples include methylation ('magic methyl'), fluorination, and bioisosteric substitution. This section explores medicinal chemistry transformations for data augmentation.

Bioisosteric Substitution

Bioisosteric substitution is the replacement of a functional group with another that has similar properties. This is commonly used in medicinal chemistry to perform different tasks such as lead optimisation (Patani and LaVoie, 1996; Langdon, Ertl and Brown, 2010).

A bioisosteric substitution operator has been developed based on the implementation used here is from Brinkmann et al. (2025). The algorithm first searched for pre-defined functional groups that have been identified to be 'chemically relevant' (these are the same set used in the functional group masking method above) (Ertl, Altmann and McKenna, 2020). All functional groups that are present have a 0.15 probability of being operated on, as defined by the authors to maximise validity. Functional groups that are selected to be operated on are replaced by randomly choosing their top-5 frequently reported bioisosteres, if any. An example is shown in Figure 8-7.

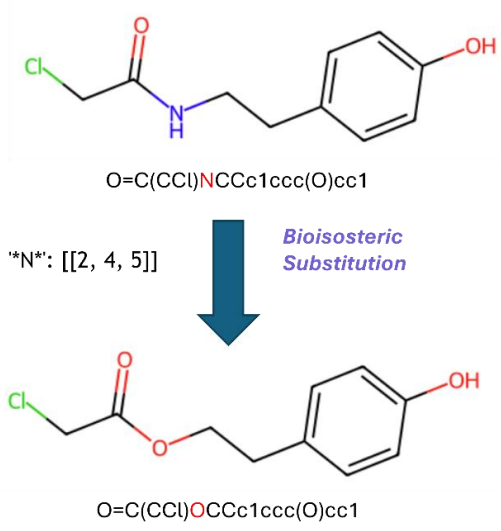


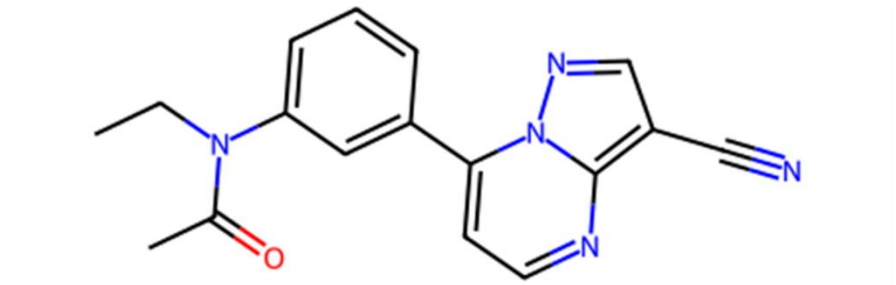
Figure 8-7. Example of Bioisosteric Substitution. The figure shows how '-NH' is replaced with a corresponding bioisostere.

8.2.5 Multimodal Fusion Operator

In the wider AI domain, multimodal data augmentation is the technique of using different modalities, such as image, text, and audio, to improve the performance of deep learning models (Zhou *et al.*, 2024). A multimodal fusion operator was developed, which randomly selects from the atom-fusion, graph-fusion, and bioisosteric substitution operators *n* times to apply to a given SMILES. This is considered as multi-modal since each operator is based on a different strategy.

8.2.6 Zaleplon MPO Task

The Zaleplon MPO task is to generate molecules that are similar to zaleplon but have a different molecular formula. The similarity between a given SMILES and zaleplon is calculated using ECFP4 fingerprints (corresponding to a Morgan fingerprint with radius = 2) (Figure 8-8). The isomer scoring function gives a lower score to SMILES that is closer to the target formula $C_{19}H_{17}N_3O_2$. Finally, the contribution of both components is averaged using the geometric mean. The Zaleplon MPO task replaced the Ranolazine MPO task from the previous chapter as it more difficult for the models to optimise.



O=C(C)N(CC)C1=CC=CC(C2=CC=NC3=C(C=NN23)C#N)=C1

Figure 8-8. Zaleplon's Structure and SMILES String Representation.

8.2.7 Experimental Setup

This investigation involved a variety of models and augmentation operators (Table 8-2). The baseline models include the RNN-LSTM and Graph-GA. The RNN-LSTM was augmented at the subset loop level using the best methods identified in the previous chapter, i.e., with 5 loops using atom-fusion 3. However, this augmentation tends to have a negative impact on the synthetic accessibility of the generated molecules. Therefore, strategies to ensure that the data augmentation leads to chemically sensible molecules were investigated. Strategies investigated include finetuning dataset filtering and substructure masking to control the augmentation operator. Additionally, graph-fusion and bioisosteric substitution were investigated as alternative forms of data augmentation operators.

All models were run three times, as described in the methods chapter, and tasked on GuacaMol's goal-directed benchmark that includes Sitagliptin MPO and Zaleplon MPO. The top-100 outputs from each model's triplicate repeats were combined and duplicates removed for the score analysis. This means that for each model, there may be fewer than 300 SMILES. The MPO, SAscore, and QED scores were averaged and plotted along with the standard deviations.

The distributions of SAscore and QED scores of molecules were also plotted for the GuacaMol training data. The GuacaMol training data contains 1,273,104 molecules. This was included to investigate how the models may deviate from this initial training data.

Table 8-2. Summary of the de novo Design Methods and Augmentation Operators.

	Method	Summary
Models	Baseline RNN-LSTM	RNN-LSTM without data augmentation.
	Baseline Graph-GA	*De novo* design algorithm that generally outperforms RNN-LSTM on GuacaMol's task.
	Augmented RNN-LSTM	RNN-LSTM augmented by operators (described below) at the subset loop stage with loops = 5.
	Filtered and Augmented RNN-LSTM	RNN-LSTM augmented by the operators, but the output SMILES is filtered prior to scoring.
Augmentation Operators	Atom-fusion Operator	NLP-inspired operator that randomly selects an atom-level operator to apply to a SMILES *n* number of times.
	Masked NLP-inspired Operators	Atom-level operators (delete, insert, and swap) that have substructure masking constraints, either with atom-level SAscore or functional group masking.
	Chemistry-based Operators	Augmentation operators that consider chemical knowledge. This includes bioisosteric substitution and the graph-level operators.
	Multimodal operator	Augmentation operator that randomly selects an operator from the above to apply to a SMILES *n* number of times.

8.3 Results and Discussion

8.3.1 SAscore and QED Analysis

Understanding the distribution of the metrics associated with chemical sense is an important consideration for understanding a *de novo* design model's behaviour. This section analyses the distributions of SAscore and QED in the input training dataset and the generated output of different models (augmented and non-augmented).

Baselines

Figure 8-9 shows the distributions of SAscore and QED for GuacaMol's training data and the output of the baseline models RNN-LSTM and Graph-GA, with the goal to optimise for the Sitagliptin and Zaleplon MPO tasks. The score distributions show binned SAscore or QED scores on the X-axis and the number of molecules within a given bin on the Y-axis. The score distributions indicate that the majority of the compounds in the GuacaMol training dataset are promising in terms of synthesisability. This is to be expected as the data is from ChEMBL, which is a curated database of known bioactive molecules with drug-like properties.

The RNN-LSTM model tasked on the Sitagliptin MPO generates a distribution similar to the SAscore distribution of the GuacaMol's training data (Figure 8-9B). For the QED distribution, the most populated bin, that is, the modal bin, for the RNN-LSTM generated molecules matches that of the GuacaMol training data (QED = 0.7-0.8). In contrast, the RNN-LSTM model tasked on the Zaleplon MPO generates a distribution with a smaller variance in terms of SAscore with a slight skew towards a lower SAscore (Figure 8-9C). However, the most populated bin matches that of the GuacaMol training data (SAscore = 2-3). Likewise, the most populated bin for the QED distribution also matches the GuacaMol training data (QED = 0.7-0.8).

On the other hand, there seem to be a difference in the performance of Graph-GAs when configured for the Sitagliptin MPO and Zaleplon MPO tasks. When tasked on the Sitagliptin MPO, there is a clear shift in distribution when comparing the output from the Graph-GA with the distributions of the input data (Figure 8-9D). The modal class of the SAscore distribution shifts from 2-3 to 5-6, whereas the QED score shifts from 0.7-0.8 to 0.5-0.6. Conversely, when tasked on the Zaleplon MPO, there is only a slight negative impact on the SAscore and QED scores (Figure 8-9E). This may suggest that the Sitagliptin MPO is more difficult for the Graph-GA.

Finally, all of the distributions generated by the baseline models generally show a sharp drop-off from the most populated bins, i.e. the mode. This is likely because the distributions for the generated molecules are calculated from a smaller sample size (i.e. 300 vs 1,273,104). As a result of the smaller sample size, there is a higher probability that the sample does not fully represent all the characteristics of the population and may not capture the full range of variability present in the population.

These results suggest that the Graph-GA is relatively poor at generating molecules that are synthesisable and drug-like when compared to the RNN-LSTM. This is surprising as the mutation algorithms of the Graph-GA were developed to create “realistic looking molecules” by following rules based on the probabilities found by analysing the ZINC dataset (Jensen, 2019). For example, the ZINC dataset has 63% of the atoms being ring atoms so the ring-insertion mutation is chosen 63% of the time in the algorithm. These results are supported by Gao and Coley (2020), who identified that the fraction of molecules generated by Graph-GA that would pass a retrosynthesis analysis tool is less than the fraction of molecules generated by an RNN-LSTM. This suggest that Graph-GA produces less synthesisable molecules.

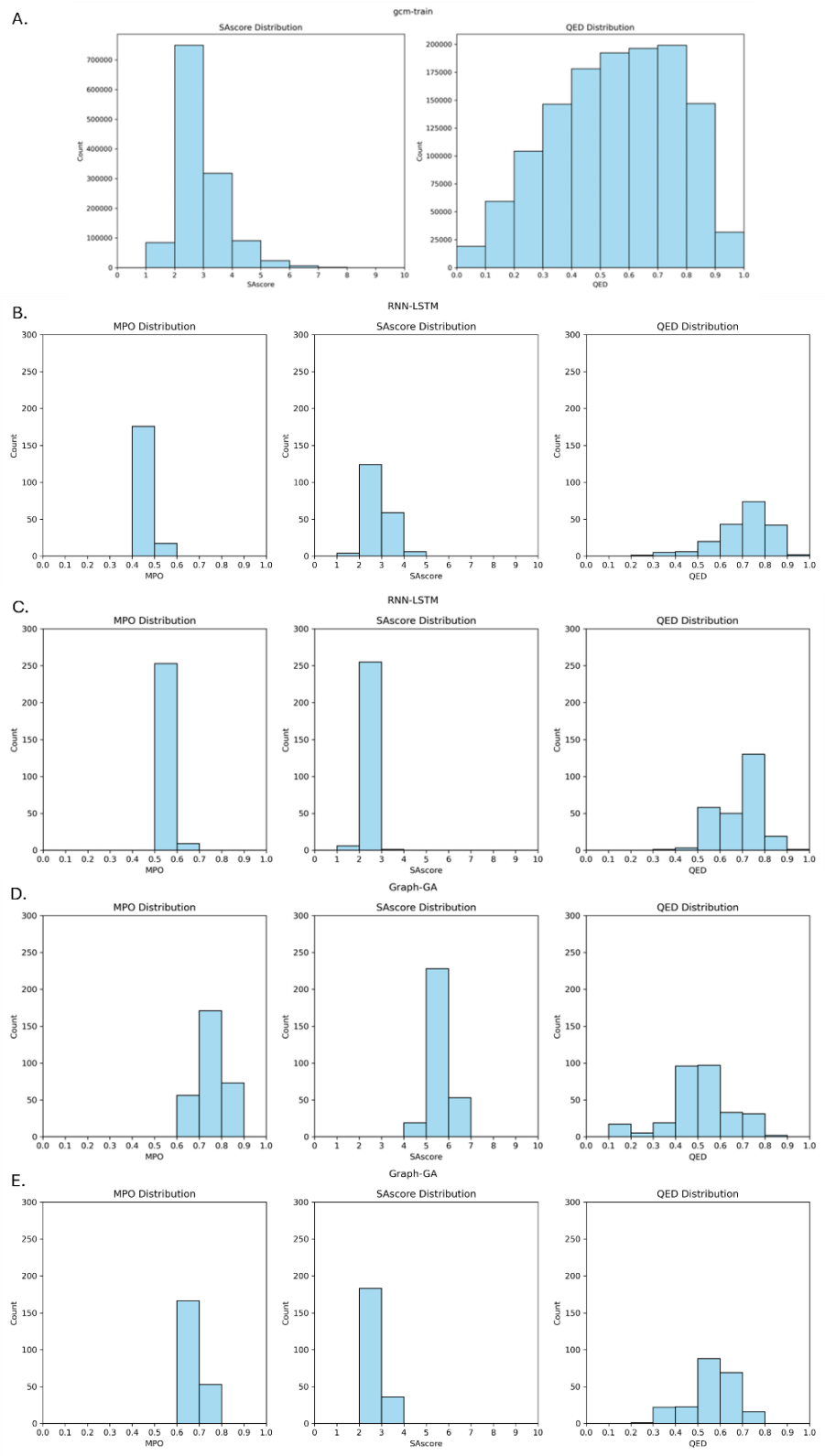


Figure 8-9. Distribution of Synthesisability-associated Metrics of the Training Data and Baseline Models. A, Distribution of the GuacaMol training data. **B,** Distribution of the output from an RNN-LSTM tasked on the Sitagliptin MPO. **C,** Distribution of the output from an RNN-LSTM tasked on the Zaleplon MPO. **D,** Distribution of the output from Graph-GA tasked on the Sitagliptin MPO. **E,** Distribution of the output from Graph-GA tasked on the Zaleplon MPO.

RNN-LSTM Augmented with Atom Fusion Operator

This section compares the MPO, SAscore, and QED scores of the molecules generated by the baselines (RNN-LSTM and Graph-GA) and the augmented RNN-LSTM models with 5 loops using the atom-fusion operator with the goal to optimise for the Sitagliptin and Zaleplon MPO tasks.

The atom-fusion operator randomly chooses an operation from the set of atom-delete, atom-insert, and atom-swap *n* number of times and applies it sequentially for a given SMILES. The process is sampling with replacement and there is therefore the chance that one type of operation may be chosen multiple times for a given SMILES. Each operator's magnitude, i.e. probability *p* of deletion or number *n* of insertions or swaps, is also randomly selected. For the atom-delete, a random probability is chosen between 0.05 and 0.15. For the atom-insert and atom-swap, a random number between 1 and 6 is chosen. Results from the previous chapter showed that the random selection of operators and magnitudes provides comparable results to the pre-determined 'greedy' approach. Additionally, random fusion is less time-consuming as prior parameter searching is not required. The approach of random operator and magnitude selection, known as automated data augmentation, is commonly used in the wider field of AI (Cubuk *et al.*, 2019).

Figure 8-10 and Appendix Table 0-1 show the average scores for the 300 molecules generated by each of the baseline (RNN-LSTM and Graph-GA) and the RNN-LSTM models augmented with atom-fusion (n = 1, 2, and 3) for the Sitagliptin MPO task across the triplicate repeats. For QED, the mean score is reduced, i.e. becomes worse, from 0.709±0.124 (RNN-LSTM baseline) to 0.457±0.145 (RNN-LSTM atom-fusion 1), 0.670±0.174 (RNN-LSTM atom-fusion 2), and 0.681+0.176 (RNN-LSTM atom-fusion 3). Unexpectedly, the mean QED score increases as the number of operations *n* per SMILES is increased using atom-fusion. A potential explanation is that more operations may help tune the properties involved in QED. A molecule that has a logP value that is too high will require the addition of polar groups or removal of non-polar groups. An atom-fusion operator with a higher number of operations may be more beneficial as it can perform more additions of polar groups and/or removals of non-polar

groups to improve the logP of the molecule. For example, atom-fusion 3 may randomly select the sequence of atom-delete, atom-insert, and atom-insert operations. This may remove a molecule's alkyl group (-CH_3) and then insert a hydroxyl group (-OH) in two different places.

In contrast, for SAscore, the mean score is increased, i.e., is worse, from 2.780±0.578 (RNN-LSTM baseline), 4.949±0.560 (RNN-LSTM atom-fusion 1), 4.848±0.495 (RNN-LSTM atom-fusion 2), and 5.160±0.534 (RNN-LSTM atom-fusion 3). These indicate the atom-fusion data augmentation method have a negative impact on the QED and SAscore. Interestingly, the output molecules from the atom-fusion operators tend to be more synthesisable than those of the Graph-GA baseline (5.626±0.380). This is surprising, as mentioned above, the Graph-GA's mutation operators were developed to create 'realistic molecules' by following heuristics identified from the analysis of ZINC dataset.

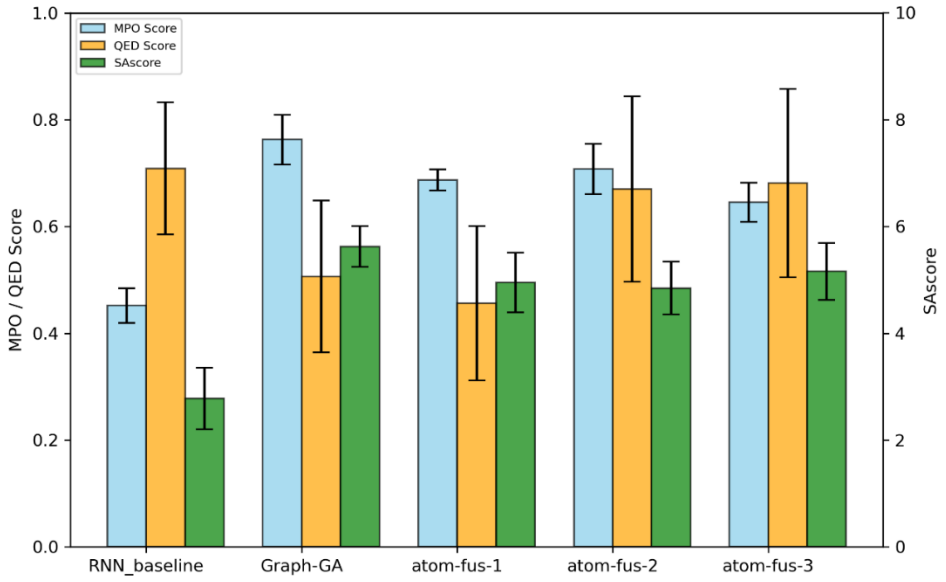


Figure 8-10. Mean Sitagliptin MPO, SAscores, and QED Scores of the Molecules Generated by the Baselines and the RNN-LSTM Models Augmented with Atom-fusion using Subset Loop = 5. The different models aimed to optimise for the sitagliptin MPO task. The results shown are averaged across three repeats. The left y-axis represents the scale of values for the MPO and QED scores. The right y-axis represents the scale of values for the SAscore.

Figure 8-11 and Appendix Table 0-3 shows the average scores for the 300 molecules each generated by the baseline (RNN-LSTM and Graph-GA) and the RNN-LSTM models augmented with atom-fusion (n = 1, 2, and 3) across the triplicate repeats for the Zaleplon MPO task. Results indicate that augmentation improves the Zaleplon MPO scores of the generated molecules, with atom-fusion 1 reaching similar performance to the Graph-GA (atom-fusion 1: 0.680±0.027 vs Graph-GA: 0.681±0.021 with p-value = ns, i.e. not significant). However, like for the Sitagliptin MPO, the atom-fusion methods have a negative impact for both QED and SAscores when compared to the RNN-LSTM baseline. Additionally, the atom-fusion operators perform worse on SAscore compared to the Graph-GA (atom-fusion 1: 3.042±0.291, atom-fusion 2: 2.855±0.125, atom-fusion 3: 2.811±0,129 vs Graph-GA: 2.774±0.248).

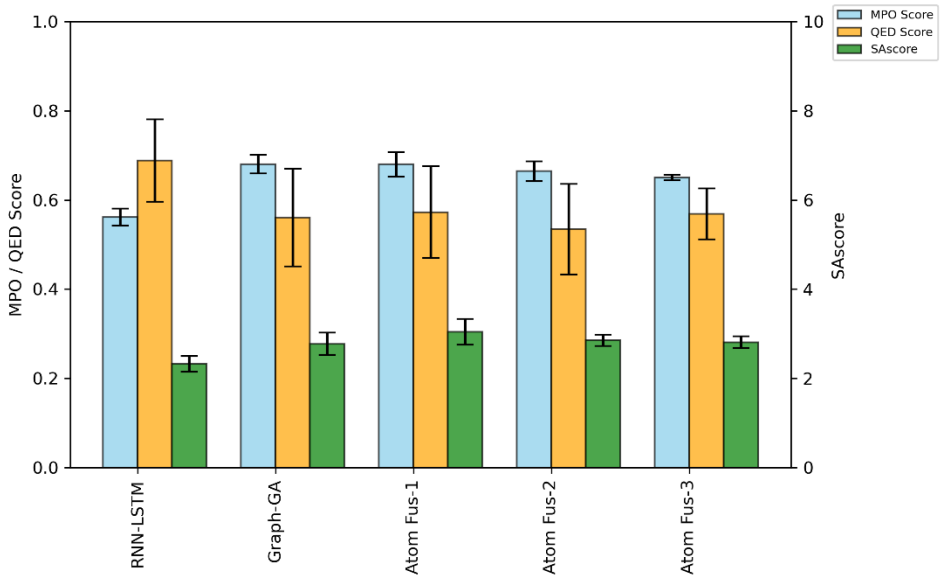


Figure 8-11. Mean Zaleplon MPO, SAscores, and QED Scores of the Molecules Generated by the Baselines and the RNN-LSTM Models Augmented with Atom-fusion using Subset Loop = 5. The different models aimed to optimise for the zaleplon MPO task. The results shown are averaged across three repeats. The left y-axis represents the scale of values for the MPO and QED scores. The right y-axis represents the scale of values for the SAscore.

The score distribution of the 300 molecules generated by the RNN-LSTMs augmented using the atom-fusion operator (n = 1, 2, and 3) with a subset loop of 5 for the Sitagliptin MPO and Zaleplon MPO tasks are shown in Figure 8-12 and Figure 8-13, respectively. The result of the Sitagliptin MPO indicate that the generated molecules tend to be less synthesisable than the RNN-LSTM baseline (Figure 8-9B). Additionally, as the number of atom-level operations *n* for atom-fusion is increased, the distribution of the SAscores shifts towards more unsynthesisable SAscores. Likewise, the results for the Zaleplon MPO indicate a skew towards worse SAscores when compared to the RNN-LSTM baseline (Figure 8-9C). Interestingly, the atom-fusion operator with n=1 has the worst SAscore distribution for the Zaleplon MPO. This was unexpected as it is assumed that more operations will lead to a worse SAscore.

These results indicate that, as expected from the previous chapter, the atom-fusion data augmentation method has a negative impact on the QED and SAscore.

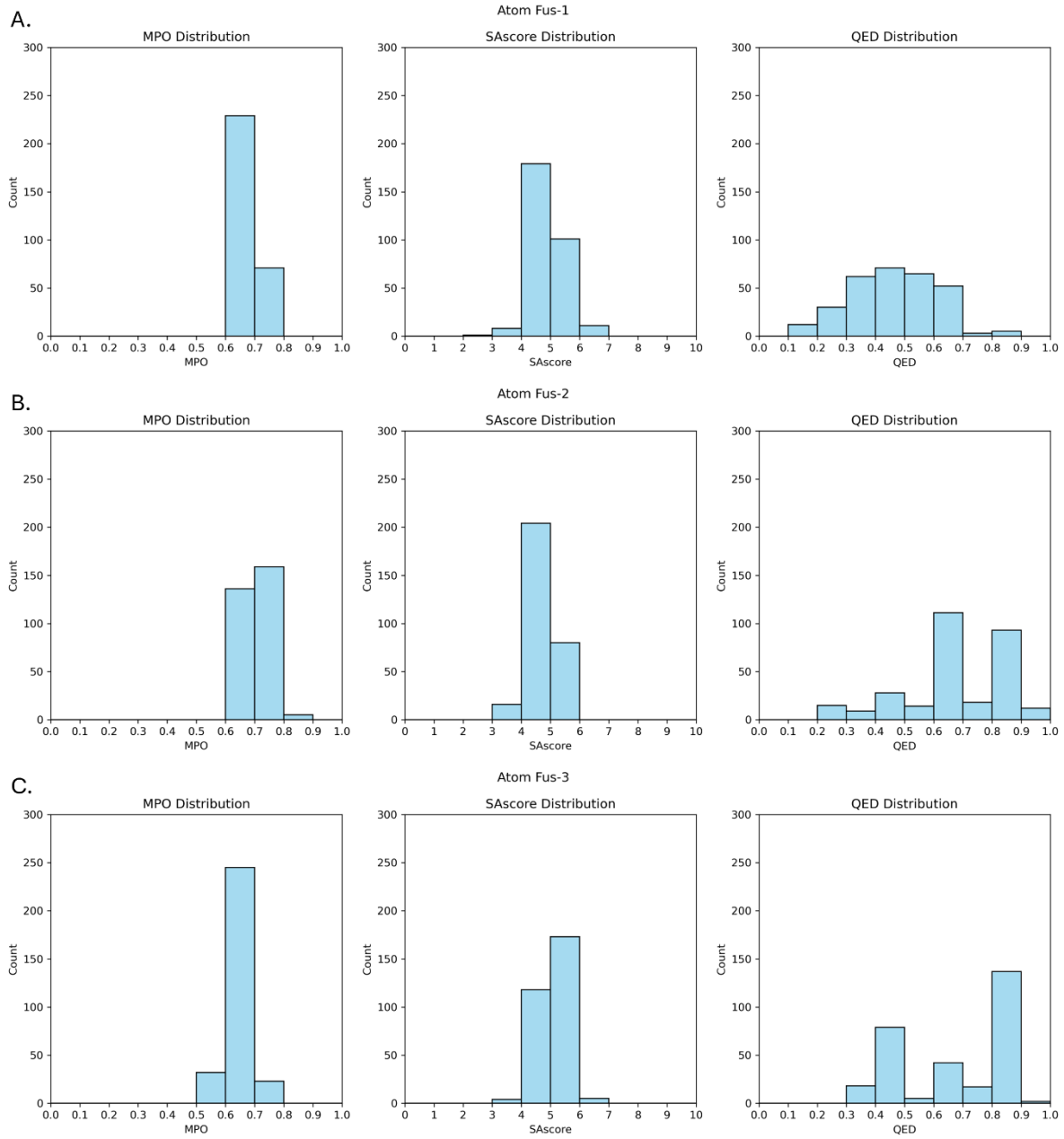


Figure 8-12. Distribution of SAscore and QED Scores for the Atom-Fusion Augmented RNN-LSTM Models for the Sitagliptin MPO Task. Distribution of the RNN-LSTM model augmented with **A.** atom-fusion 1 operator with subset loop of 5. **B.** atom-fusion 2 operator with subset loop of 5. **C.** atom-fusion 3 operator with subset loop of 5.

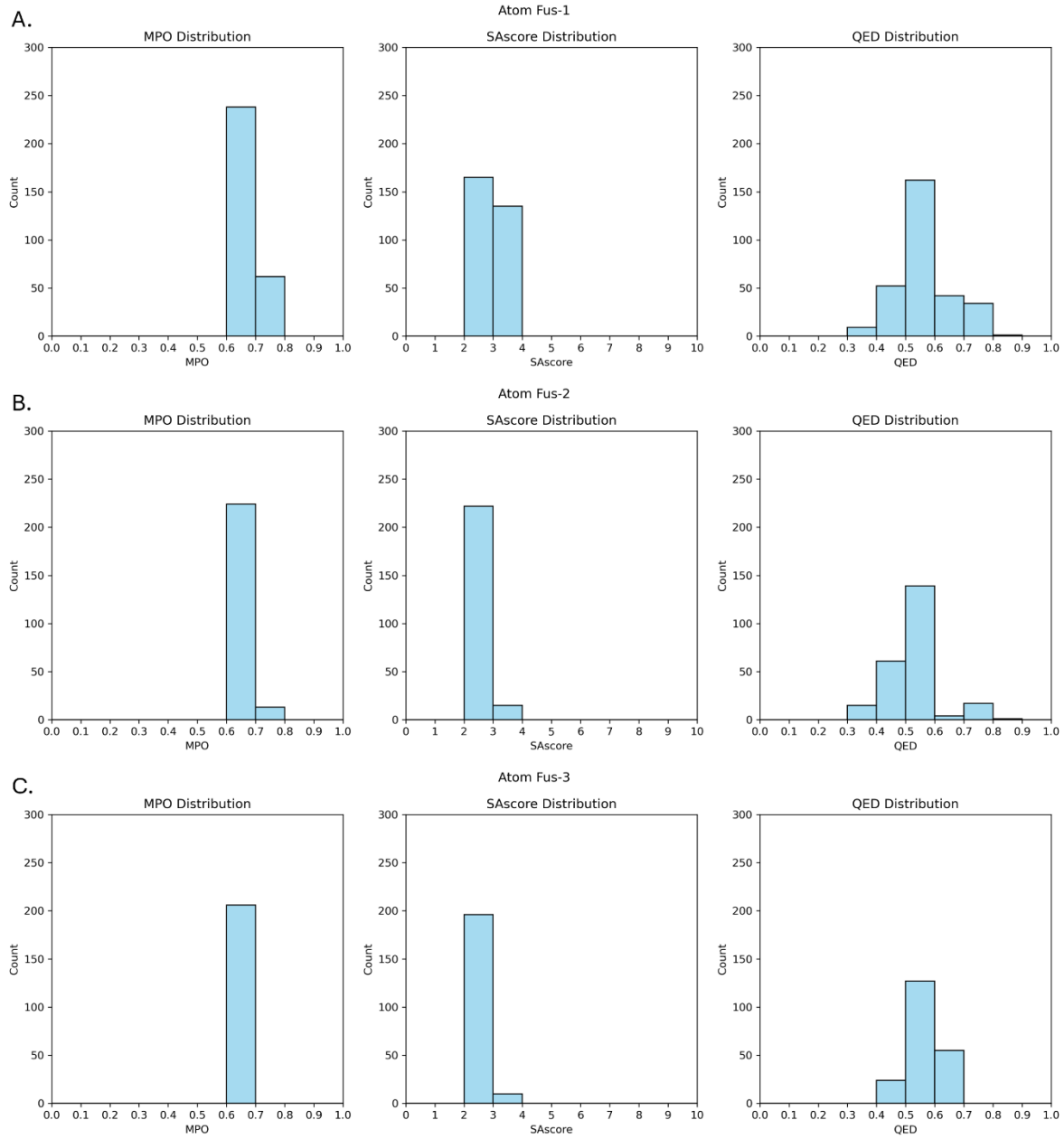


Figure 8-13. Distribution of SAscore and QED Scores for the Atom-Fusion Augmented RNN-LSTM Models for the Zaleplon MPO Task. Distribution of the RNN-LSTM model augmented with **A.** atom-fusion 1 operator with subset loop of 5. **B.** atom-fusion 2 operator with subset loop of 5. **C.** atom-fusion 3 operator with subset loop of 5.

8.3.2 Substructure Masking

Masking important substructures is a potential way to limit the negative effect of the atom-level operators on synthesisability. This section presents the results for the atom SAscore-based masking and functional group masking. SAscore-based masking (with thresholds of 0, 1, and 2) and functional group masking were investigated. The SAscore threshold prevents an operator from editing atoms with SAscore higher than a user-defined threshold, as measured by the atom SAscore described earlier. Whereas the functional group mask is based on a predefined set of 'chemically-relevant' SMARTS patterns.

Atom-Delete Masking

Table 8-3 and Appendix Figure 0-1 shows the result of RNN-LSTM models augmented using the atom-delete operator (with probability of deletion p = 0.15) with and without masking for the Sitagliptin MPO task. Results indicate that the SAscore-based masking with threshold = 0 improved the SAscore of the generated molecules compared to performing atom-delete without masking (2.940±0.676 vs 3.379±0.428 with p <= 0.0001). However, masking with thresholds = 1 and 2 have some negative impact on the generated molecules' SAscores (3.531±0.791 with p <= 0.05 and 3.477±0.551 with p <= 0.05, respectively). This is likely because fewer atoms are masked at higher thresholds. Therefore, atoms with atom-level SAscores within the ranges 0 – 2, which are considered 'favourable', will have been modified by the operator and lead to an overall decrease in the molecule's SAscore. In contrast, FG masking improved the SAscore of the generated molecules compared to performing atom-delete without masking (2.853±0.544 vs 3.379±0.428 with p <= 0.0001).

On the other hand, Table 8-4 and Appendix Figure 0-2 show the results when the models are tasked on the Zaleplon MPO. Results indicate that masking with SAscore did not have a significant impact on the SAscore with the exception of the functional group masking (2.622±0.279 with p <= 0.01). These results may be because the augmentation without masking is already producing molecules with low SAscores.

In terms of the MPO score, introducing masking generally had a negative impact on the Sitagliptin and Zaleplon scores of the generated molecules. However, the results of these masks are still better compared to the molecules generated by the RNN baseline without augmentation for Sitagliptin MPO (0.452±0.033) and Zaleplon MPO (0.562±0.019).

Table 8-3. The Sitagliptin MPO, SAscore, and QED Scores of the Molecules Generated by the RNN-LSTM Augmented Using Atom-Delete With and Without Masking. SAS mask refers to the SAscore-based masking at different thresholds. The atom-delete augmentation was performed with a probability p = 0.15 at the subset loop = 5 point. The results are averaged across the 300 molecules generated across triplicate repeats. Results in bold indicate a significantly improved score compared to the augmentation without masking. The p-value was obtained using t-test against the augmentation without masking. The full result for the statistical testing is in Appendix Table 0-5. The p-value indicates levels of significance: **** (p <= 0.0001), *** (p <= 0.001), ** (p <= 0.01), * (p <= 0.05), and ns (p > 0.05).

Model	MPO	p-value	QED	p-value	SAscore	p-value
Atom Delete-015	0.552±0.030	-	0.638±0.185	-	3.379±0.428	-
Atom Delete-015 SAS mask 0	0.464±0.038	****	**0.702±0.150**	****	**2.940±0.676**	****
Atom Delete-015 SAS mask 1	0.474±0.029	****	**0.680±0.185**	*	3.531±0.791	*
Atom Delete-015 SAS mask 2	**0.585±0.033**	****	0.594±0.170	**	3.477±0.551	*
Atom Delete-015 FG mask	0.486±0.042	****	**0.670±0.159**	*	**2.853±0.544**	****

Table 8-4. The Zaleplon MPO, SAscore, and QED Scores of the Molecules Generated by the RNN-LSTM Augmented Using Atom-Delete With and Without Masking. SAS mask refers to the SAscore-based masking at different thresholds. The atom-delete augmentation was performed with a probability p = 0.15 at the subset loop = 5 point. The results are averaged across the 300 molecules generated across triplicate repeats. Results in bold indicate a significantly improved score compared to the augmentation without masking. The p-value was obtained using t-test against the augmentation without masking. The full result for the statistical testing is in Appendix Table 0-6. The p-value indicates levels of significance: **** (p <= 0.0001), *** (p <= 0.001), ** (p <= 0.01), * (p <= 0.05), and ns (p > 0.05).

Model	MPO	p-value	QED	p-value	SAscore	p-value
Atom Delete-015	0.629±0.023	-	0.572±0.083	-	2.683±0.327	-
Atom Delete-015 SAS mask 0	0.617±0.019	****	0.573±0.081	ns	2.631±0.278	ns
Atom Delete-015 SAS mask 1	0.618±0.017	****	0.543±0.074	****	2.702±0.302	ns
Atom Delete-015 SAS mask 2	0.622±0.016	***	0.551±0.067	**	2.720±0.341	ns
Atom Delete-015 FG mask	0.627±0.022	ns	0.572±0.090	ns	**2.622±0.279**	**

Atom-Swap Masking

Table 8-5 and Appendix Figure 0-3 show the result of RNN-LSTM models augmented using the atom-swap operator (number of swaps *n* = 6) with and without masking for the Sitagliptin MPO task. Results generally show no significant differences between masking and without masking. An exception is when masking using functional group-based masking, which improved the MPO scores of the generated molecules.

On the other hand, Table 8-6 and Appendix Figure 0-4 shows the results when the augmented models are tasked on the Zaleplon MPO. Results generally indicate no significant differences on the SAscore, an exception is the functional group-based masking which made the SAscore worse. Interestingly, the use of masking with thresholds 0 and 2 improved the Zaleplon MPO scores.

Table 8-5. The Sitagliptin MPO, SAscore, and QED Scores of the Molecules Generated by the RNN-LSTM Augmented Using Atom-Swap With and Without Masking. SAS mask refers to the SAscore-based masking at different thresholds. The atom-swap augmentation was performed with swap number *n* = 6 at the subset loop = 5 point. The results are averaged across the 300 molecules generated across triplicate repeats. Results in bold indicate a significantly improved score compared to the augmentation without masking. The p-value was obtained using t-test against the augmentation without masking. The full result for the statistical testing is in Appendix Table 0-7. The p-value indicates levels of significance: **** (p <= 0.0001), *** (p <= 0.001), ** (p <= 0.01), * (p <= 0.05), and ns (p > 0.05).

Model	MPO	p-value	QED	p-value	SAscore	p-value
Atom Swap-6	0.564±0.027	-	0.631±0.149	-	4.238±0.456	-
Atom Swap-6 SAS mask 0	0.567±0.028	ns	0.620±0.139	ns	4.247±0.497	ns
Atom Swap-6 SAS mask 1	0.561±0.023	ns	0.655±0.133	ns	4.190±0.550	ns
Atom Swap-6 SAS mask 2	0.561±0.025	ns	0.614±0.148	ns	4.242±0.527	ns
Atom Swap-6 FG mask	**0.575±0.031**	****	0.591±0.139	**	4.224±0.543	ns

Table 8-6. The Zaleplon MPO, SAscore, and QED Scores of the Molecules Generated by the RNN-LSTM Augmented Using Atom-Swap With and Without Masking. SAS mask refers to the SAscore-based masking at different thresholds. The atom-swap augmentation was performed with swap number *n* = 6 at the subset loop = 5 point. The results are averaged across the 300 molecules generated across triplicate repeats. Results in bold indicate a significantly improved score compared to the augmentation without masking. The p-value was obtained using t-test against the augmentation without masking. The full result for the statistical testing is in Appendix Table 0-8. The p-value indicates levels of significance: **** (p <= 0.0001), *** (p <= 0.001), ** (p <= 0.01), * (p <= 0.05), and ns (p > 0.05).

Model	MPO	p-value	QED	p-value	SAscore	p-value
Atom Swap-6	0.668±0.044	-	0.536±0.108	-	2.843±0.303	-
Atom Swap-6 SAS mask 0	**0.702±0.015**	****	0.523±0.118	ns	2.830±0.202	ns
Atom Swap-6 SAS mask 1	0.667±0.041	ns	**0.560±0.122**	*	2.873±0.238	ns
Atom Swap-6 SAS mask 2	**0.700±0.008**	****	0.528±0.113	ns	2.814±0.120	ns
Atom Swap-6 FG mask	0.654±0.040	***	0.507±0.114	**	2.943±0.332	***

Atom-Fusion Masking

Table 8-7 and Appendix Figure 0-5 show the result of RNN-LSTM models augmented using the atom-fusion operator (number of atom-operators *n* = 3) with and without masking tasked on the Sitagliptin MPO. Results indicate that only the SAscore-based masking with threshold = 0 improved the SAscore compared to no masking (4.380±0.532 vs 4.946±0.459 with p <= 0.01). However, this comes at the expense of the MPO score (0.643±0.032 vs 0.652±0.040 with p <= 0.01).

On the other hand, Table 8-8 and Appendix Figure 0-6 shows the result when the RNN-LSTM model augmented using the atom fusion operator (number of atom-operators *n* = 1) are tasked on the Zaleplon MPO. Result indicates some improvement when masking, with the functional group-based masking performing the best (2.806±0.163 vs 2.905±0.182 with p<=0.0001).

Table 8-7. The Sitagliptin MPO, SAscore, and QED Scores of the Molecules Generated by the RNN-LSTM Augmented Using Atom-Fusion With and Without Masking. SAS mask refers to the SAscore-based masking at different thresholds. The atom-fusion augmentation was performed with the number of random atom-operators *n* = 3 at the subset loop = 5 point. The results are averaged across the 300 molecules generated across triplicate repeats. Results in bold indicate a significantly improved score compared to the augmentation without masking. The p-value was obtained using t-test against the augmentation without masking. The full result for the statistical testing is in Appendix Table 0-9. The p-value indicates levels of significance: **** (p <= 0.0001), *** (p <= 0.001), ** (p <= 0.01), * (p <= 0.05), and ns (p > 0.05).

Model	MPO	p-value	QED	p-value	SAscore	p-value
Atom Fusion 3	0.652±0.040	-	0.757±0.129	-	4.496±0.459	-
Atom Fusion 3 SAS mask 0	0.643±0.032	**	0.656±0.156	****	**4.380±0.532**	**
Atom Fusion 3 SAS mask 1	0.639±0.032	****	0.677±0.152	****	4.608±0.619	*
Atom Fusion 3 SAS mask 2	0.637±0.033	****	0.702±0.141	****	4.690±0.503	****
Atom Fusion 3 FG mask	**0.675±0.038**	****	0.759±0.143	ns	4.741±0.414	****

Table 8-8. The Zaleplon MPO, SAscore, and QED Scores of the Molecules Generated by the RNN-LSTM Augmented Using Atom-Fusion With and Without Masking. SAS mask refers to the SAscore-based masking at different thresholds. The atom-fusion augmentation was performed with the number of random atom-operators *n* = 1 at the subset loop = 5 point. The results are averaged across the 300 molecules generated across triplicate repeats. Results in bold indicate a significantly improved score compared to the augmentation without masking. The p-value was obtained using t-test against the augmentation without masking. The full result for the statistical testing is in Appendix Table 0-10. The p-value indicates levels of significance: **** (p <= 0.0001), *** (p <= 0.001), ** (p <= 0.01), * (p <= 0.05), and ns (p > 0.05).

Model	MPO	p-value	QED	p-value	SAscore	p-value
Atom Fusion 1	0.695±0.039	-	0.553±0.118	-	2.905±0.182	-
Atom Fusion 1 SAS mask 0	0.678±0.023	****	0.574±0.096	*	2.978±0.348	**
Atom Fusion 1 SAS mask 1	0.668±0.024	****	0.541±0.088	ns	**2.850±0.202**	**
Atom Fusion 1 SAS mask 2	0.669±0.021	****	0.525±0.107	**	**2.816±0.143**	****
Atom Fusion 1 FG mask	0.656±0.017	****	0.549±0.093	ns	**2.806±0.163**	****

Discussion

The results of this section suggest that using functional group-based masking performed better compared to the SAscore-based masking. The functional group-based masking improved the SAscore for atom-delete and atom-swap tasked on the Sitagliptin MPO. When tasked on the Zaleplon MPO, it improved the SAscore for atom-delete, and atom-fusion.

The SAscore-based masking may not perform as well because it may not consider important substructures as a whole. Within a substructure, there is likely to be a range of atom-level SAscores. For example, some atoms may have an atom-level SAscore in the range of 0-2 and some have an atom-level SAscore >2. Having a masking threshold of 2 may mean that some atoms within the substructure will be edited, disrupting the substructure and therefore lead to a negative impact on the overall SAscore of the molecule. For example, in Figure 8-4, the phenol ring contains atom-level SAscore that ranges from 1.5-2.5. This may explain why a functional group mask generally performs better because it considers atoms together in the context of a substructure rather than atom-by-atom.

8.3.3 Chemistry-based Data Augmentation

8.3.3.1 Graph-based Operation

Molecules are more naturally represented as graphs rather than as SMILES strings and, therefore, performing operations in the graph representation level may be chemically more meaningful and help maintain/improve the synthesisability during augmentation. This section compares the scores of the molecules generated by the baselines (RNN-LSTM and Graph-GA) and the augmented RNN-LSTM models with 5 loops using the graph-fusion operator with the goal to optimise for the MPO tasks.

The graph-fusion operator randomly chooses a graph mutation *n* number of times and applies it sequentially for a given molecule. The choice of graph mutations is from the implementation of Jensen (2019). Like the atom-fusion operator above, the process is sampling with replacement and there is the chance that one type of operation may be chosen multiple times for a given molecule.

Figure 8-14 and Appendix Table 0-11 show the average scores for the 300 molecules each generated by the baseline (RNN-LSTM and Graph-GA) and the RNN-LSTM models augmented with graph-fusion (n = 1, 2, and 3) for the Sitagliptin MPO task across the triplicate repeats. Results indicate that the RNN-LSTM augmented with graph-fusions (n=1: 0.867±0.021, n=2: 0.823±0.028, and n=3 0.799±0.024) outperformed both the RNN-LSTM (0.452±0.033) and Graph-GA (0.763±0.046) baselines in terms of the MPO score.

Figure 8-15 and Appendix Table 0-13 show the average scores for the 300 molecules each generated by the baseline (RNN-LSTM and Graph-GA) and the RNN-LSTM models augmented with graph-fusion (n = 1, 2, and 3) for the Zaleplon MPO task across the triplicate repeats. Results indicate that the RNN-LSTM augmented with graph-fusions (n=1: 0.714±0.016, n=2: 0.715±0.016, and n=3 0.707±0.011) outperformed both the RNN-LSTM (0.562±0.019) and Graph-GA (0.681±0.021) baselines in terms of the MPO score.

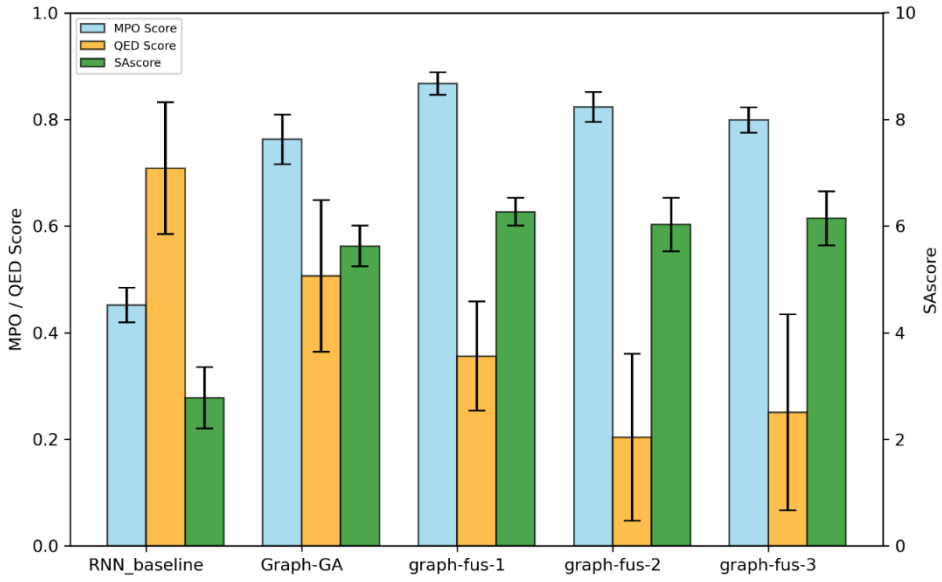


Figure 8-14 Mean Sitagliptin MPO, SAscore, and QED Scores of the Molecules Generated by the Baselines and the RNN-LSTM Models Augmented with Graph-Fusion Using Subset Loop = 5. The different models aimed to optimise for the sitagliptin MPO task. The results shown are averaged across three repeats. The left y-axis represents the scale of values for the MPO and QED scores. The right y-axis represents the scale of values for the SAscore.

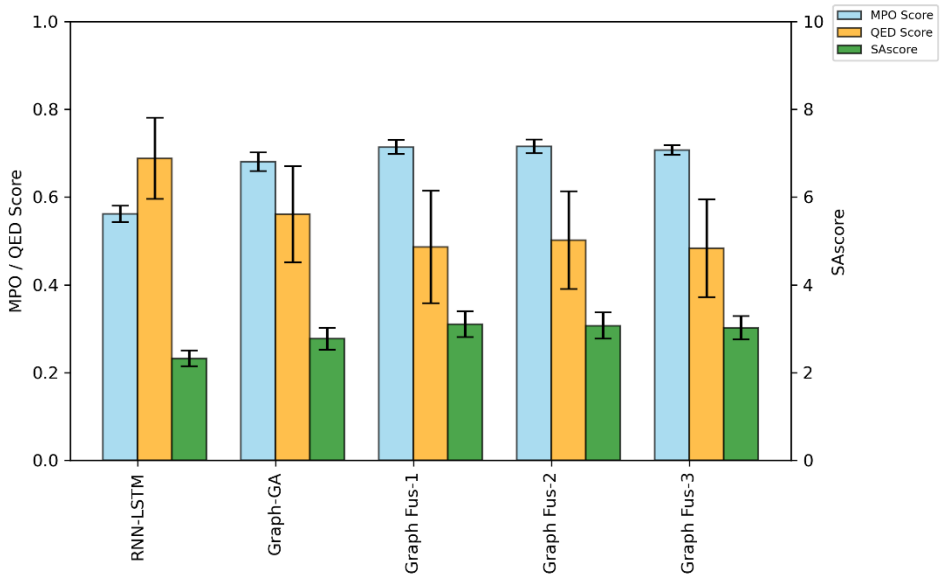


Figure 8-15 Mean Zaleplon MPO, SAscore, and QED Scores of the Molecules Generated by the Baselines and the RNN-LSTM Models Augmented with Graph-Fusion Using Subset Loop = 5. The different models aimed to optimise for the sitagliptin MPO task. The results shown are averaged across three repeats. The left y-axis represents the scale of values for the MPO and QED scores. The right y-axis represents the scale of values for the SAscore.

The SAscore and QED score distributions of the molecules generated by the RNN-LSTM augmented with the graph-fusion operators using subset loop = 5 were then investigated for the Sitagliptin MPO and Zaleplon MPO task, as shown in Figure 8-16 and Figure 8-17, respectively. Results indicate a distribution shift towards a higher SAscores, and hence worse synthesisability, and lower QED scores when compared to the RNN-LSTM baselines (Figure 8-9) and the RNN-LSTM augmented with atom-fusion (Figure 8-12 and Figure 8-13) for both Sitagliptin MPO and Zaleplon MPO Tasks.

A worst result compared to the RNN-LSTM baseline was expected as results above have shown that the Graph-GA generates molecules that are worse in terms of SAscores and QED scores. However, what is surprising is that the combination of the RNN-LSTM and the graph-fusion augmentation performing worse than the Graph-GA. The expectation was that the combination should have generated molecules that are on the 'middle-ground' of the RNN-LSTM and graph mutation.

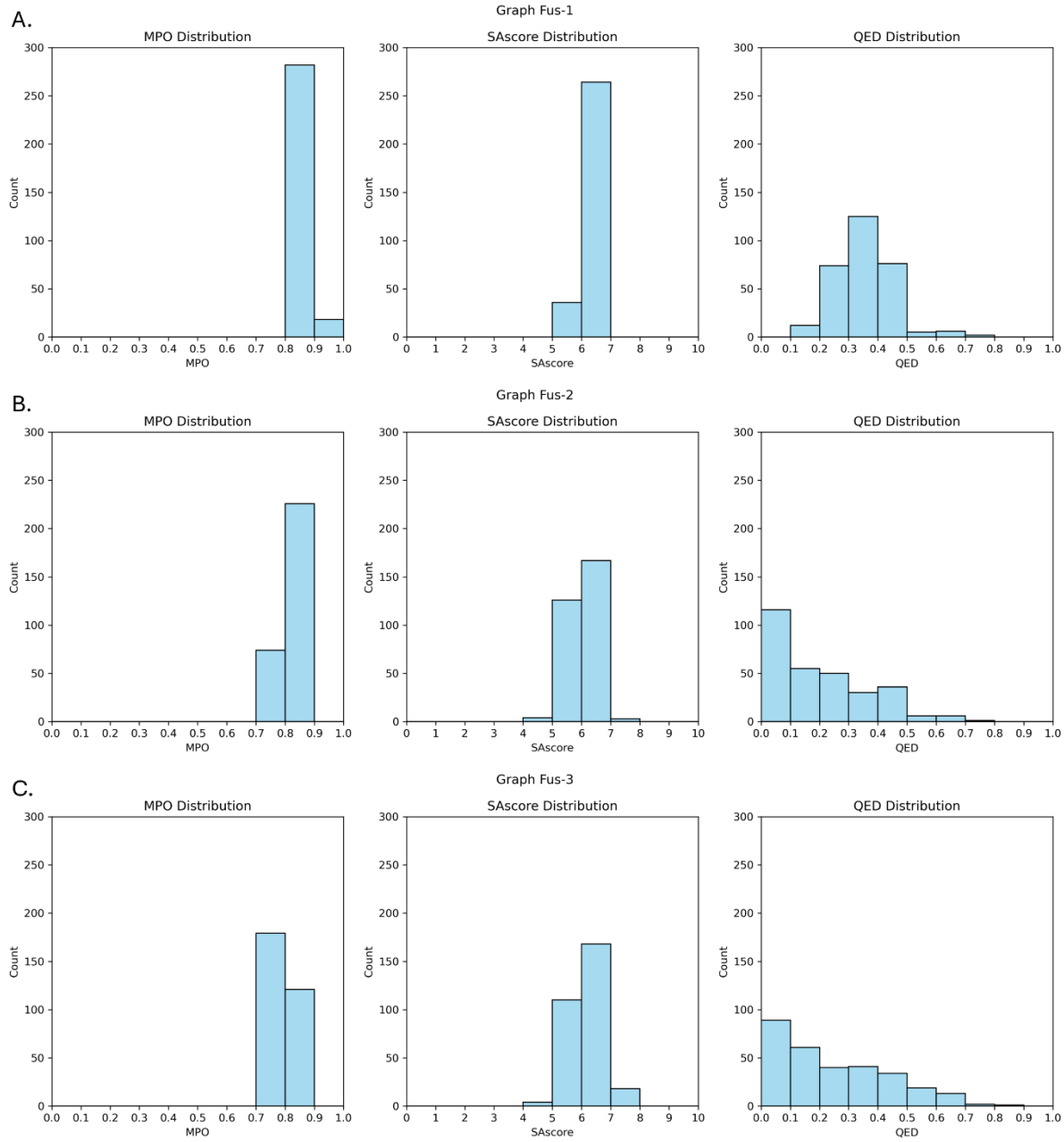


Figure 8-16. Distribution of the Sitagliptin MPO, SAscore, and QED Scores of the Molecules Generated by the RNN-LSTM Augmented with Graph-Fusions Using a Subset Loop = 5. A, Graph-Fusion n = 1 with subset loop = 5. **B,** Graph-Fusion n = 2 with subset loop = 5. **C,** Graph-Fusion n = 1 with subset loop = 5. The molecules were generated across three repeats, with duplicated being removed.

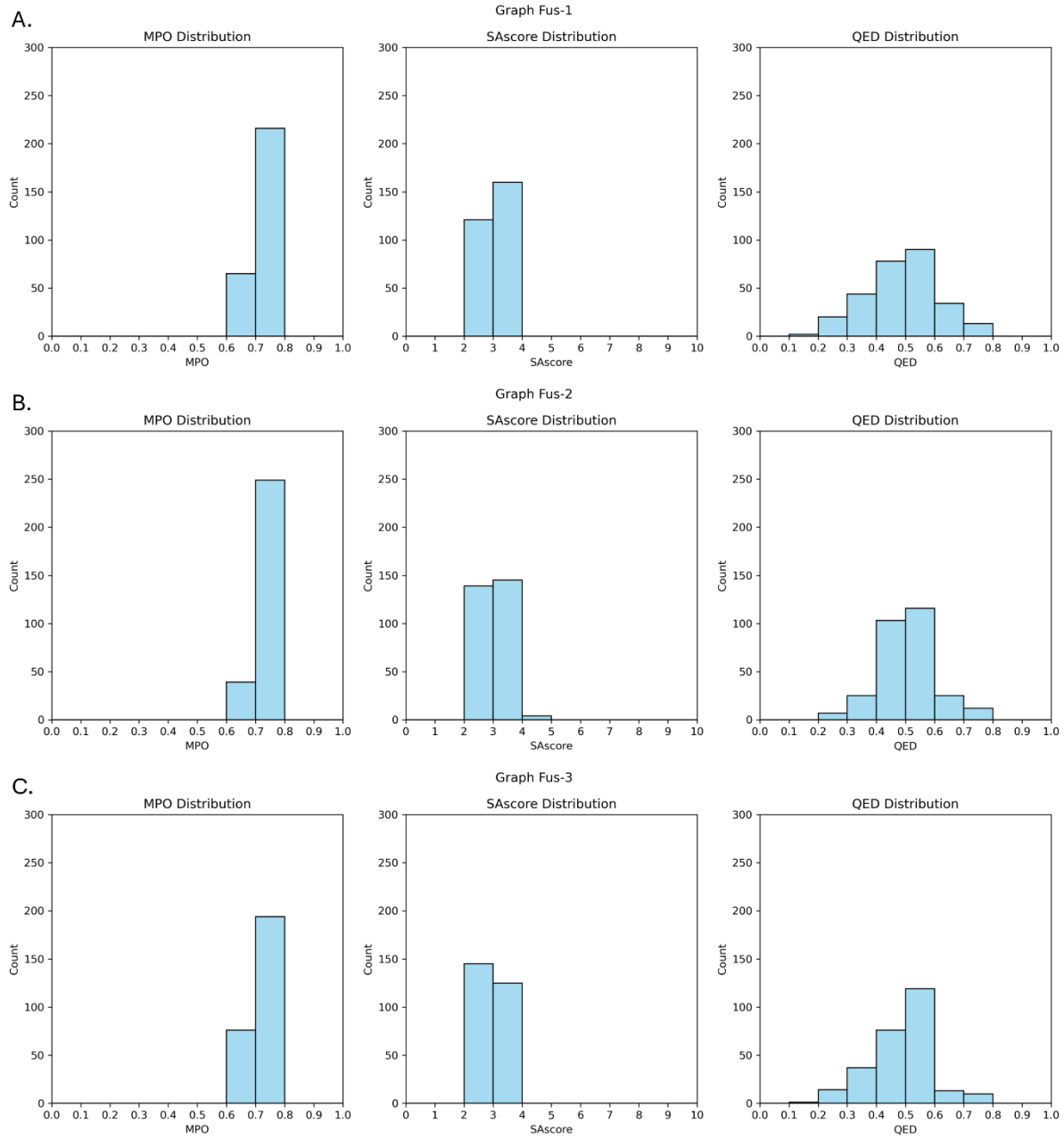


Figure 8-17. Distribution of the Zaleplon MPO, SAscore, and QED of the Molecules Generated by the RNN-LSTM Augmented with Graph-Fusions Using a Subset Loop = 5. A, Graph-Fusion n = 1 with subset loop = 5. **B,** Graph-Fusion n = 2 with subset loop = 5. **C,** Graph-Fusion n = 1 with subset loop = 5. The molecules were generated across three repeats, with duplicated being removed.

8.3.3.2 Medicinal Chemistry Transformation

Bioisosteric Substitution

Table 8-9 and Appendix Figure 0-7 show the result of the RNN-LSTM models augmented using different operators (atom-fusion 3, graph-fusion 1, and bioisosteric substitution) tasked on the Sitagliptin MPO. Results indicate bioisosteric substitution has an improved SAscore of (3.019±0.294) compared to the other augmentation operators, atom-fusion 3 with functional group-based masking (4.741±0.414) and graph-fusion 1 (6.2690±0.260). However, the MPO score for the graph-fusion 1 operator still performs the best (0.867±0.021) compared to the bioisosteric substitution (0.615±0.021) or the atom-fusion 3 with functional group-based masking (0.675±0.038) operators.

Table 8-10 and Appendix Figure 0-8 show the result of the RNN-LSTM models augmented using different operators (atom-fusion 1, graph-fusion 1, and bioisosteric substitution) tasked on the Zaleplon MPO. Results indicates that the atom fusion 1 with functional group-based masking improved the SAscores the most (2.806±0.163) compared to the other augmentation operators, graph fusion 1 (3.106±0.295) and bioisosteric substitution (2.968±0.446). Interestingly, the Graph-GA performed better than the augmentation operators on the SAscores in contrast to the results in the Sitagliptin MPO.

Table 8-9. The Sitagliptin MPO, SAscore, and QED Scores of the Molecules Generated by the Baselines (RNN-LSTM and Graph-GA) and the RNN-LSTM Augmented with Different Operators (Atom-Fusion 3, Graph-Fusion 1, and Bioisosteric Substitution) at the Subset Loop = 5 Point. The results are averaged across the 300 molecules generated across triplicate repeats Results in bold indicate a significantly improved score compared to the baselines. The p-value was obtained using a pairwise comparison t-test against the RNN-LSTM, pv (p-value) RNN, and Graph-GA, pv (p-value) GA. The full result for the statistical testing is in Appendix Table 0-15. The p-value indicates levels of significance: **** (p <= 0.0001), *** (p <= 0.001), ** (p <= 0.01), * (p <= 0.05), and ns (p > 0.05).

Model	MPO	pv RNN	pv GA	QED	pv RNN	pv GA	SAscore	pv RNN	pv GA
RNN-LSTM	0.452±0.033	-	****	0.709±0.124	-	****	2.780±0.577	-	****
Graph-GA	**0.763±0.046**	****	-	0.506±0.142	****	-	5.626±0.380	****	-
Atom Fusion 3 FG mask	**0.675±0.038**	****	****	**0.759±0.143**	****	****	4.741±0.414	****	****
Graph Fus-1	**0.867±0.021**	****	****	0.356±0.103	****	****	6.269±0.260	****	****
Bioisosteric Substitution	**0.615±0.027**	****	****	0.573±0.164	****	****	3.019±0.294	****	****

Table 8-10. The Zaleplon MPO, SAscore, and QED Scores of the Molecules Generated by the Baselines (RNN-LSTM and Graph-GA) and the RNN-LSTM Augmented with Different Operators (Atom-Fusion 1 FG Mask, Graph-Fusion 1, and Bioisosteric Substitution) at the Subset Loop = 5 Point. The results are averaged across the 300 molecules generated across triplicate repeats. Results in bold indicate a significantly improved score compared to the baselines. The p-value was obtained using a pairwise comparison t-test against the RNN-LSTM, pv (p-value) RNN, and Graph-GA, pv (p-value) GA. The full result for the statistical testing is in Appendix Table 0-16. The p-value indicates levels of significance: **** (p <= 0.0001), *** (p <= 0.001), ** (p <= 0.01), * (p <= 0.05), and ns (p > 0.05).

Model	MPO	pv RNN	pv GA	QED	pv RNN	pv GA	SAscore	pv RNN	pv GA
RNN-LSTM	0.562±0.019	-	****	0.688±0.092	-	****	2.323±0.179	-	****
Graph-GA	**0.681±0.021**	****	-	0.561±0.110	****	-	2.774±0.248	****	-
Atom Fusion 1 FG mask	**0.656±0.017**	****	****	0.549±0.093	****	****	2.806±0.163	****	****
Graph Fus-1	**0.714±0.016**	****	****	0.486±0.128	****	****	3.106±0.295	****	****
Bioisosteric Substitution	**0.657±0.024**	****	****	0.625±0.092	****	****	2.968±0.446	****	****

8.3.4 Multimodal Fusion Operator

The proposed operators (atom-fusion, graph-fusion, and bioisosteric substitution) have shown to improve the MPO score of the generated molecules when augmenting the RNN-LSTM model. However, they have shown to have a negative impact on the QED score and SAscores of the generated molecules. This section explores whether combining the different operators, into what is termed here as a 'multimodal' operator, will improve the SAscores of the generated molecules by averaging out the differences of the operators. The multimodal fusion operator introduced here randomly selects an operation *n* times from the set of atom-fusion 1, graph-fusion 1, and bioisosteric substitution for a given SMILES.

Table 8-11 and Appendix Figure 0-9 shows the average Sitagliptin MPO, SAscore and QED score of the SMILES generated by the RNN-LSTM augmented with the multimodal fusion operators, *n* = 1 and 2, compared to the model baselines and the RNN-LSTM augmented with the previously developed operators. Results indicate that the use of multimodal fusion 1 is able to improve the SAscore when compared to graph-fusion 1, which is the augmentation operator that has the most negative impact on the SAscore (5.776±0.492 vs 6.269±0.260 with p-value < 0.0001 Appendix Table 0-17). However, the remains significantly worse compared to the baseline RNN-LSTM.

In contrast, the multimodal fusion operator does not seem to have as much of a benefit when used to augment the RNN-LSTM in the Zaleplon MPO task (Table 8-12 and Appendix Figure 0-10)**.** However, the multimodal fusion operators were able to outperform the other augmentation operators in terms of the MPO scores.

Table 8-11. The Sitagliptin MPO, SAscore, and QED Scores of the Molecules Generated by the Baselines (RNN-LSTM and Graph-GA) and the RNN-LSTM Augmented with Different Operators (Atom-Fusion 1 FG Mask, Graph-Fusion 1, Bioisosteric Substitution, and Multimodal-Fusions) at the Subset Loop = 5 Point. The results are averaged across the 300 molecules generated across triplicate repeats. Results in bold indicate a significantly improved score compared to the baselines. The p-value was obtained using a pairwise comparison t-test against the RNN-LSTM, pv (p-value) RNN, and Graph-GA, pv (p-value) GA. The full result for the statistical testing is in Appendix Table 0-17. The p-value indicates levels of significance: **** ($p <= 0.0001$), *** ($p <= 0.001$), ** ($p <= 0.01$), * ($p <= 0.05$), and ns ($p > 0.05$).

Model	MPO	pv RNN	pv GA	QED	pv RNN	pv GA	SAscore	pv RNN	pv GA
RNN-LSTM	0.452±0.033	-	****	0.709±0.124	-	****	2.780±0.577	-	****
Graph-GA	**0.763±0.046**	****	-	0.506±0.142	****	-	5.626±0.380	****	-
Atom Fusion 3 FG mask	**0.675±0.038**	****	****	**0.759±0.143**	****	****	4.741±0.414	****	****
Graph Fus-1	**0.867±0.021**	****	****	0.356±0.103	****	****	6.269±0.260	****	****
Bioisosteric Substitution	**0.615±0.027**	****	****	0.573±0.164	****	****	3.019±0.294	****	****
Multimodal Fusion 1	**0.776±0.021**	****	****	0.421±0.173	****	****	5.776±0.492	****	****
Multimodal Fusion 2	**0.778±0.025**	****	****	0.305±0.172	****	****	6.066±0.411	****	****

Table 8-12. The Zaleplon MPO, SAscore, and QED Scores of the Molecules Generated by the Baselines (RNN-LSTM and Graph-GA) and the RNN-LSTM Augmented with Different Operators (Atom-Fusion 1 FG Mask, Graph-Fusion 1, Bioisosteric Substitution, and Multimodal-Fusions) at the Subset Loop = 5 Point. The results are averaged across the 300 molecules generated across triplicate repeats. Results in bold indicate a significantly improved score compared to the baselines. The p-value was obtained using a pairwise comparison t-test against the RNN-LSTM, pv (p-value) RNN, and Graph-GA, pv (p-value) GA. The full result for the statistical testing is in Appendix Table 0-18. The p-value indicates levels of significance: **** (p <= 0.0001), *** (p <= 0.001), ** (p <= 0.01), * (p <= 0.05), and ns (p > 0.05).

Model	MPO	pv RNN	pv GA	QED	pv RNN	pv GA	SAscore	pv RNN	pv GA
RNN-LSTM	0.562±0.019	-	****	0.688±0.092	-	****	2.323±0.179	-	****
Graph-GA	**0.681±0.021**	****	-	0.561±0.110	****	-	2.774±0.248	****	-
Atom Fusion 1 FG mask	**0.656±0.017**	****	****	0.549±0.093	****	ns	2.806±0.163	****	ns
Graph Fus-1	**0.714±0.016**	****	****	0.486±0.128	****	****	3.106±0.295	****	****
Bioisosteric Substitution	**0.657±0.024**	****	****	0.625±0.092	****	****	2.968±0.446	****	****
Multimodal Fusion 1	**0.734±0.014**	****	****	0.576±0.119	****	ns	3.137±0.381	****	****
Multimodal Fusion 2	**0.719±0.013**	****	****	0.525±0.116	****	***	3.042±0.301	****	****

8.3.5 Finetuning Dataset Filtering

The previous section has shown that the proposed augmentation operators have a negative impact on the SAscores and QED scores of the generated molecules. This is an issue for real world drug discovery applications as any molecules proposed by a generative model will need to be synthesised to be validated and used. Therefore, the next set of experiments aimed to improve the synthesisability of the output molecules through the use of filtering. The different filters used originated from different organisations and are applied individually to understand their impact on the generation process.

RNN-LSTM Baseline

Table 8-13 and Appendix Figure 0-11 show the average Sitagliptin MPO, SAscore, and QED of the SMILES generated by the RNN-LSTM baseline without and with filtering during the top-k selection process. Results generally indicate no significant differences for the average MPO, SAscore, and QED scores. In contrast, Inpharmatica, LINT and SureChEMBL improved the SAscore of the generated molecules when used to filter the RNN-LSTM baseline for the Zaleplon MPO task (Table 8-14 and Figure 0-12).

Figure 8-18 and Figure 8-19 shows the total proportion of molecules that passed through the different filters during the top-k selection of the RNN-LSTM baseline when tasked on the Sitagliptin MPO and Zaleplon MPO, respectively. Results indicate that the Dundee, Inpharmatica, LINT, and MLSMR filtered out more of the generated molecules. This may suggest that these filters are more stringent than the BMS, Glaxo, PAINS and SureChEMBL.

Table 8-13. The Sitagliptin MPO, SAscore, and QED Scores of the Molecules Generated by the RNN-LSTM Baseline Without and With Filtering. The results are averaged across the 300 molecules generated across triplicate repeats. Results in bold indicate a significantly improved score compared to the baseline without filtering. The p-value was obtained using t-test against the baseline without filtering. The full result for the statistical testing is in Appendix Table 0-19. The p-value indicates levels of significance: **** (p <= 0.0001), *** (p <= 0.001), ** (p <= 0.01), * (p <= 0.05), and ns (p > 0.05).

Model	MPO	p-value	QED	p-value	SAscore	p-value
RNN-LSTM	0.452±0.033	-	0.709±0.124	-	2.780±0.577	-
BMS	0.454±0.038	ns	0.714±0.129	ns	2.796±0.605	ns
Dundee	0.433±0.039	****	0.729±0.110	ns	2.886±0.575	ns
Glaxo	0.448±0.031	ns	0.723±0.118	ns	2.880±0.557	ns
Inpharmatica	0.455±0.044	ns	0.737±0.110	*	2.858±0.572	ns
LINT	0.446±0.033	ns	0.723±0.122	ns	2.746±0.530	ns
MLSMR	0.428±0.036	****	0.735±0.107	*	2.911±0.533	*
PAINS	0.450±0.037	ns	0.691±0.145	ns	2.758±0.582	ns
SureChEMBL	0.450±0.034	ns	0.729±0.118	ns	2.811±0.602	ns

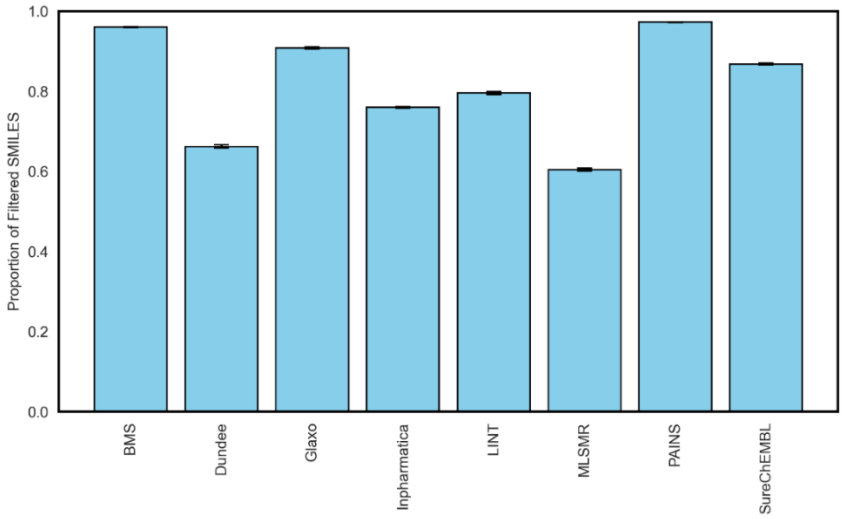


Figure 8-18. Proportion of SMILES that Passed the Filters During the Top-k Selection in the RNN Baseline Tasked on the Sitagliptin MPO. The results shown are averaged across three repeats.

Table 8-14. The Zaleplon MPO, SAscore, and QED Scores of the Molecules Generated by the RNN-LSTM Baseline Without and With Filtering. The results are averaged across the 300 molecules generated across triplicate repeats. Results in bold indicate a significantly improved score compared to the baseline without filtering. The p-value was obtained using t-test against the baseline without filtering. The full result for the statistical testing is in Appendix Table 0-20. The p-value indicates levels of significance: **** (p <= 0.0001), *** (p <= 0.001), ** (p <= 0.01), * (p <= 0.05), and ns (p > 0.05).

Model	MPO	p-value	QED	p-value	SAscore	p-value
RNN-LSTM	0.562±0.019	-	0.688±0.092	-	2.323±0.179	-
BMS	0.566±0.018	*	0.692±0.078	ns	2.300±0.165	ns
Dundee	**0.568±0.017**	***	0.700±0.086	ns	2.336±0.180	ns
Glaxo	0.558±0.018	*	0.694±0.090	ns	2.296±0.174	ns
Inpharmatica	0.562±0.022	ns	0.690±0.078	ns	**2.267±0.161**	***
LINT	0.558±0.018	*	**0.711±0.081**	**	**2.292±0.182**	*
MLSMR	0.560±0.017	ns	**0.721±0.067**	****	2.331±0.195	ns
PAINS	0.563±0.018	ns	0.698±0.077	ns	2.291±0.185	*
SureChEMBL	0.559±0.014	ns	**0.708±0.080**	**	**2.272±0.158**	***

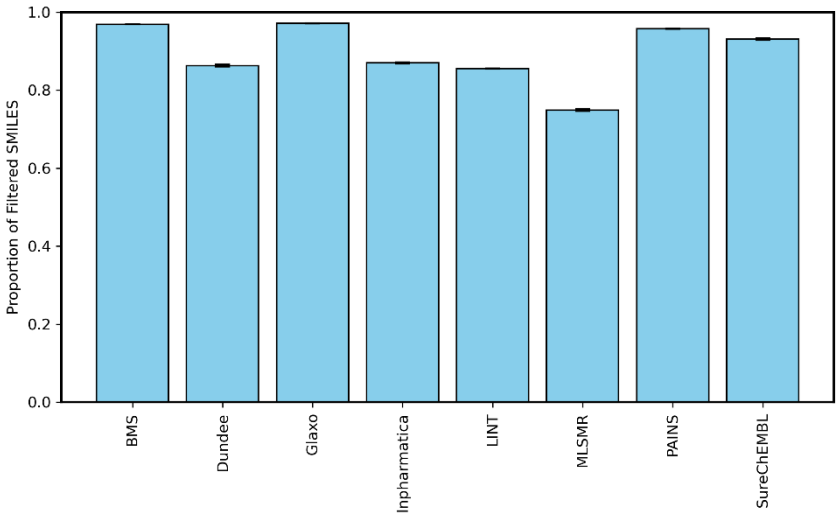


Figure 8-19. Proportion of SMILES that Passed the Filters During the Top-k Selection in the RNN Baseline Tasked on the Zaleplon MPO. The results shown are averaged across three repeats.

Different filters had different effect on the MPO scores, QED scores, and SAscores of the molecules generated by the RNN-LSTM baseline. A few filters did have a negative effect on the scores of the molecules generated during the Sitagliptin MPO task. On the other hand, a few filters had a positive effect on the scores of the molecules generated for the Zaleplon MPO task. However, the majority of the filters had no significant effect on the scores. This is likely because RNN-LSTM generates molecules that follow the data distribution of the molecules it was pretrained on, i.e. the ChEMBL dataset which contains bioactive molecules with drug-like properties and fragments that are considered desirable and synthesised often. Therefore, the model will generate molecules that will likely not be flagged by the filters' alerts.

RNN-LSTM with Data Augmentations – Sitagliptin MPO

Table 8-15 and Appendix Figure 0-13 show the average Sitagliptin MPO, SAscore, and QED scores of the molecules generated by the RNN-LSTM augmented with atom-fusion 3 at the subset loop = 5 without and with filtering. Results indicate that the introduction of filtering before scoring and storing the augmented SMILES improved the SAscore and QED scores of the generated molecules. However, this comes at the expense of the Sitagliptin MPO score. The exception to this is the LINT and PAINS filters which improved both the SAscore and the MPO Scores but no difference in the QED score.

Table 8-15. The Sitagliptin MPO, SAscore, and QED Scores of the Molecules Generated by the RNN-LSTM augmented with Atom-Fusion 3 at the Subset Loop = 5 Point Without and With Filtering. The results are averaged across the 300 molecules generated across triplicate repeats. Results in bold indicate a significantly improved score compared to the atom-fusion 3 without filtering. The p-value was obtained using t-test against the atom-fusion 3 without filtering. The full result for the statistical testing is in Appendix Table 0-21. The p-value indicates levels of significance: **** ($p <= 0.0001$), *** ($p <= 0.001$), ** ($p <= 0.01$), * ($p <= 0.05$), and ns ($p > 0.05$).

Model	MPO	p-value	QED	p-value	SAscore	p-value
Atom Fus-3	0.645±0.037	-	0.681±0.176	-	5.160±0.535	-
BMS	0.631±0.036	****	**0.787±0.107**	****	**4.509±0.422**	****
Dundee	0.636±0.026	***	**0.725±0.070**	****	**4.228±0.331**	****
Glaxo	0.645±0.036	ns	**0.718±0.148**	**	**4.750±0.426**	****
Inpharmatica	0.578±0.011	****	**0.742±0.042**	****	**3.921±0.221**	****
LINT	**0.673±0.061**	****	0.702±0.099	ns	**4.518±0.497**	****
MLSMR	0.632±0.032	****	**0.758±0.101**	****	**4.365±0.392**	****
PAINS	**0.666±0.048**	****	0.687±0.101	ns	**4.361±0.380**	****
SureChEMBL	0.627±0.025	****	**0.718±0.071**	***	**4.199±0.232**	****

Table 8-16 and Appendix Figure 0-14 show the average Sitagliptin MPO, SAscore, and QED Scores of the molecules generated by the RNN-LSTM augmented with graph-fusion 1 at the subset loop = 5 without and with filtering. Like the results for atom-fusion 3 above, filtering generally improved the SAscore and QED scores of the generated molecules, but this comes at the cost of the Sitagliptin MPO score. Exceptions to this are the Glaxo and PAINS filters, which did not change the QED score significantly. In contrast, LINT did not change the SAscore significantly.

Table 8-16. The Sitagliptin MPO, SAscore, and QED Scores of the Molecules Generated by the RNN-LSTM augmented with Graph-Fusion 1 at the Subset Loop = 5 Point Without and With Filtering. The results are averaged across the 300 molecules generated across triplicate repeats. Results in bold indicate a significantly improved score compared to the graph-fusion 1 without filtering. The p-value was obtained using t-test against the graph-fusion 1 without filtering. The full result for the statistical testing is in Appendix Table 0-22. The p-value indicates levels of significance: **** ($p <= 0.0001$), *** ($p <= 0.001$), ** ($p <= 0.01$), * ($p <= 0.05$), and ns ($p > 0.05$).

Model	MPO	p-value	QED	p-value	SAscore	p-value
Graph Fus-1	0.867±0.021	-	0.356±0.103	-	6.269±0.260	-
BMS	0.832±0.038	****	0.332±0.154	*	**6.118±0.510**	****
Dundee	0.830±0.033	****	**0.762±0.057**	****	**5.846±0.328**	****
Glaxo	0.837±0.016	****	0.369±0.099	ns	6.316±0.279	*
Inpharmatica	0.765±0.041	****	0.325±0.138	**	**5.888±0.394**	****
LINT	0.743±0.034	****	**0.576±0.157**	****	6.228±0.338	ns
MLSMR	0.840±0.020	****	**0.461±0.182**	****	**6.154±0.382**	****
PAINS	0.858±0.024	****	0.348±0.129	ns	6.376±0.230	****
SureChEMBL	0.823±0.027	****	**0.532±0.143**	****	**6.028±0.279**	****

RNN-LSTM with Data Augmentations – Zaleplon MPO

Table 8-17 and Appendix Figure 0-15 show the average Zaleplon MPO, SAscore, and QED scores of the molecules generated by the RNN-LSTM augmented with atom-fusion 1 at the subset loop = 5 point without and with filtering. Results indicate that filtering generally improved the SAscore of the generated molecules, with the LINT filter producing the best SAscore (2.762±0.133). However, this comes at the cost of reducing the MPO score.

Table 8-17. The Zaleplon MPO, SAscore, and QED Scores of the Molecules Generated by the RNN-LSTM Augmented With Atom-Fusion 1 at the Subset loop = 5 point Without and With Filtering. The results are averaged across the 300 molecules generated across triplicate repeats. Results in bold indicate a significantly improved score compared to the atom-fusion 1 without filtering. The p-value was obtained using t-test against the atom-fusion 3 without filtering. The full result for the statistical testing is in Appendix Table 0-23. The p-value indicates levels of significance: **** (p <= 0.0001), *** (p <= 0.001), ** (p <= 0.01), * (p <= 0.05), and ns (p > 0.05).

Model	MPO	p-value	QED	p-value	SAscore	p-value
Atom Fus-1	0.680±0.027	-	0.573±0.103	-	3.042±0.291	-
BMS	0.686±0.049	ns	0.593±0.092	*	**2.850±0.226**	****
Dundee	0.650±0.007	****	0.579±0.044	ns	**2.780±0.128**	****
Glaxo	0.663±0.020	****	0.569±0.100	ns	**2.783±0.168**	****
Inpharmatica	0.643±0.007	****	0.559±0.053	*	**2.823±0.142**	****
LINT	0.650±0.007	****	0.577±0.041	ns	**2.762±0.133**	****
MLSMR	0.645±0.012	****	0.568±0.074	ns	**2.857±0.198**	****
PAINS	0.646±0.009	****	0.542±0.074	****	**2.783±0.159**	****
SureChEMBL	0.671±0.034	**	**0.589±0.075**	*	**2.967±0.241**	**

Table 8-18 and Appendix Figure 0-16 show the average Zaleplon MPO, SAscore, and QED scores of the SMILES generated by the RNN-LSTM augmented with graph-fusion 1 at the subset loop = 5 point without and with filtering. Unlike the result for filtering of atom-fusion 1 for the Zaleplon MPO task, filtering has less of an impact on the SAscores of the generated molecules when using graph-fusion 1 for data augmentation. Here, the BMS filter performs the best (3.043±0.327). Filtering here also has a negative impact on the MPO score.

Table 8-18. The Zaleplon MPO, SAscore, and QED Scores of the Molecules Generated by the RNN-LSTM Augmented With Graph-Fusion 1 at the Subset Loop = 5 Point Without and With Filtering. The results are averaged across the 300 molecules generated across triplicate repeats. Results in bold indicate a significantly improved score compared to the graph-fusion 1 without filtering. The p-value was obtained using t-test against the atom-fusion 3 without filtering. The full result for the statistical testing is in Appendix Table 0-24. The p-value indicates levels of significance: **** (p <= 0.0001), *** (p <= 0.001), ** (p <= 0.01), * (p <= 0.05), and ns (p > 0.05).

Model	MPO	p-value	QED	p-value	SAscore	p-value
Graph Fus-1	0.714±0.016	-	0.486±0.128	-	3.106±0.295	-
BMS	0.705±0.012	****	0.480±0.106	ns	**3.034±0.363**	*
Dundee	0.677±0.012	****	**0.680±0.075**	****	3.270±0.327	****
Glaxo	0.702±0.010	****	0.469±0.103	ns	**3.010±0.257**	****
Inpharmatica	0.693±0.013	****	**0.512±0.098**	**	3.109±0.255	ns
LINT	0.688±0.015	****	**0.573±0.123**	****	3.107±0.339	ns
MLSMR	0.672±0.015	****	**0.583±0.115**	****	3.252±0.315	****
PAINS	**0.726±0.028**	****	0.495±0.119	ns	**3.042±0.267**	**
SureChEMBL	0.664±0.016	****	**0.606±0.111**	****	3.233±0.413	****

Discussion

The results in this section show that introducing filtering in the augmentation process generally helps the augmented RNN-LSTM models to generate molecules with an improved SAscore. Although the filtering has a negative impact on the MPO score, the MPO score is still higher than the RNN-baseline without filtering and data augmentation. Interestingly, for both the Sitagliptin and Zaleplon MPO tasks, all types of filters improved the SAscores of the molecules produced when using the atom-fusion operator. In contrast, not all filter types improved the SAscores of the molecules produced when using the graph-fusion operator. A potential explanation may be that the graph-operator produce rarer substructures. These may not be identified by the filters as they are based on known problematic substructures.

8.4 Conclusion

This chapter explored how the synthesisability when performing data augmentation for generative *de novo* design can be improved. The first investigation involved using substructure mask to improve the atom-level operators proposed from the previous chapter. The next investigation involved applying operators that use more chemical knowledge to data augmentation for generative *de novo* design. Finally, the finetuning dataset filtering was investigated.

The first set of results show that although the NLP-inspired operators improved the MPO score of the generated molecules, they tend to have a negative impact on the SAscores of the generated molecules. This motivated the development of the masking and chemistry-based operators to address this issue. Results indicate that the performance of these alternative operators may be task-dependent. However, bioisosteric substitution may be the best in general as it is able to significantly increase the MPO scores but only slightly decrease the SAscore of the generated molecules. Finally, results show that the finetuning dataset filtering provide a more general benefit on the SAscores of the molecules generated by the augmented RNN-LSTM models configured on both the Sitagliptin MPO and Zaleplon tasks. Although these led to a reduction in the MPO scores of the molecules, it is still higher compared to the MPO scores of the molecules generated from the baseline RNN-LSTM. This supports the finding that filtering is a viable strategy for filtering potentially 'unsynthesisable' motifs, but this may come at the cost of a lower objective function score (Gao and Coley, 2020).

A limitation of this chapter is that the data augmentation methods are applied to tasks from the GuacaMol benchmark, which does not fully reflect how drug discovery is applied in real projects. Therefore, the next chapter investigates the effect of the operators on more realistic case studies.

Chapter 9. Case Study

9.1 Introduction

The rapid development of generative *de novo* design models has given rise to benchmarks to standardise evaluation and allow for comparison between models, such as the GuacaMol benchmark which has been extensively used in the previous chapters (Brown *et al.*, 2019). However, recent studies have highlighted potential limitations of these benchmarks and raises the question of whether they are applicable to practical drug discovery, as discussed in the literature review (Renz *et al.*, 2019; Grant and Sit, 2021; Tripp and Hernández-Lobato, 2023).

This chapter aims to evaluate generative *de novo* design models with more realistic methods of evaluation. The first evaluation uses QSAR models as a goal-directed generation task. QSAR has the advantage of simulating biological or physicochemical interactions, in contrast to GuacaMol's similarity-based metrics, and represents a more realistic task in drug discovery. The QSAR models are integrated into GuacaMol's RNN-LSTM model to perform goal-directed generation and investigate how performing data augmentation affects model performance. Secondly, the ability of the model to generate diverse and novel structures is analysed using Bemis-Murcko scaffolds. This post-hoc analysis is also commonly used in real drug-discovery projects, for example, to perform scaffold hopping where the aim is to identify compounds with desired properties but which belong to different chemical series to those already known.

9.2 Methods

9.2.1 MolScore

MolScore is an open-source platform for benchmarking *de novo* design models available at https://github.com/MorganCThomas/MolScore (Thomas *et al.*, 2024). The benchmark provides quantitative structure-activity relationship (QSAR) models, such as PIDGINv5 which is described below, that have been trained on relevant bioactivity data to evaluate *de novo* design models. The MolScore benchmark was therefore used in this chapter to provide a more realistic assessment by evaluating how generated molecules meet specific biological objectives, in contrast to the GuacaMol benchmark tasks investigated in previous chapters which rely on similarity-based metrics.

9.2.1.1 PIDGINv5

The PIDGINv5 framework is a collection of 2337 pretrained random forest classification models trained to predict the probability of activity of an input molecule against a panel of membrane proteins (available at https://doi.org/10.5281/zenodo.7547691). The training workflow is described in Mervin et al. (2015). Briefly, the training dataset consists of molecules extracted from the ChEMBL31 and PubChem databases with known activities against specific protein targets. The molecules were converted to 2048-bit Morgan circular fingerprints, i.e. ECFP4, using RDKit, to capture relevant substructures. Finally, the models were trained on these fingerprints and they output predicted probabilities of bioactivities against a target at specified concentrations, e.g. 1µM and 10 µM. The predicted probabilities are derived from the average output of individual decision trees within the RF model, i.e. the proportion of decision trees that predict if a compound is active.

5-HT_{2A} Selectivity Benchmark

The 5-HT_{2A} selectivity benchmark is a collection of tasks centred on maximising activity against the 5-HT_{2A} receptor whilst minimising off-target bioactivity. The 5-HT_{2A} receptor is a subtype of serotonin receptors involved in various neuropsychological processes and is a therapeutically important target for antipsychotics. This chapter focuses on the 5-HT_{2A} tasks as a single objective problem and 5-HT_{2A} combined with the Serotonin and Dopamine tasks as a multiobjective problem.

In the 5-HT_{2A} task, the model aims to generate molecules that have a maximised predicted probability of activity against a 5-HT_{2A} target receptor at a 1µM concentration. For the combined 5-HT_{2A}, Serotonin, and Dopamine task, the model aims to generate molecules that have a maximised predicted probability of activity against a 5-HT_{2A} target receptor at a 1µM concentration whilst minimising activity against off-target receptors that belong in the serotonin and dopamine receptor families at a 10µM concentration.

Table 9-1 shows how the final 5-HT_{2A}, Serotonin, and Dopamine MPO score is calculated in MolScore for the COCCOC(=O)c1cc(C(=O)c2cc(F)ccc2O)coc1=N SMILES as an example. The first step is to calculate the predicted probability of activity against target receptors of interests using PIDGINv5's QSAR models. The 5-HT_{2A} has one, Serotonin family has ten, and Dopamine family has six target receptor QSAR models respectively. The different QSAR models are denoted by their UniProt accession ID. The output probabilities within each family of targets are averaged to give the family average probability. The next step is to transform the family average probabilities. The 5-HT_{2A} family average probability is unaltered as the aim is to maximise activity against this family. In contrast, the Serotonin and Dopamine families average probabilities are transformed such that a high probability of activity is given a low score, and a low probability of activity is given a high score as the aim is to minimise activity against these families. This performed by subtracting the values from , e.g. the normalised Serotonin average probability = 0.984 = 1 – 0.016. Finally, the MPO is calculated as the arithmetic mean of the normalised family average probabilities/scores.

Table 9-1. Example Calculation of the MPO for the 5-HT$_{2A}$, Serotonin, and Dopamine Task.

Receptor Family	Target	Predicted Probability/Score
5HT2A	P28223@1uM	0.021
	Family average probability	0.021
Serotonin	P08908@10uM	0.000
	P28221@10uM	0.014
	P28222@10uM	0.014
	P28335@10uM	0.000
	P30939@10uM	0.091
	P34969@10uM	0.015
	P41595@10uM	0.030
	P47898@10uM	0.000
	P50406@10uM	0.000
	Q13639@10uM	0.000
	Family average probability	0.016
Dopamine	P14416@10uM	0.000
	P21728@10uM	0.000
	P21917@10uM	0.006
	P21918@10uM	0.042
	P35462@10uM	0.000
	Family average probability	0.010
	normalised 5HT2A average	0.021
	normalised Serotonin average	0.984
	normalised Dopamine average	**0.990**
	Final MPO	**0.665**

9.2.2 Bemis-Murcko Scaffold Novelty

A Bemis-Murcko (BM) scaffold represents a molecule's core structure (Bemis and Murcko, 1996). BM scaffold is generated by removing the side-chain atoms found in a molecule to leave only ring systems and the linker atoms that connect the rings. BM scaffold analysis is an important consideration in realistic drug design as it can indicate the ability of a *de novo* design model/program to explore the chemical space in a diverse manner.

Figure 9-1 show the workflow for the BM analysis workflow. The first step is to clean the molecules, both the GuacaMol training dataset and the generated molecules, by using RDKit's sanitisation and canonicalisation functions to ensure reproducibility. RDKit's MurckoScaffold function is then used to extract BM scaffolds from the molecules. Unique scaffolds indicate distinct scaffolds found in the generated molecules. Novel scaffolds indicate generated scaffolds not already found in the original GuacaMol training dataset.

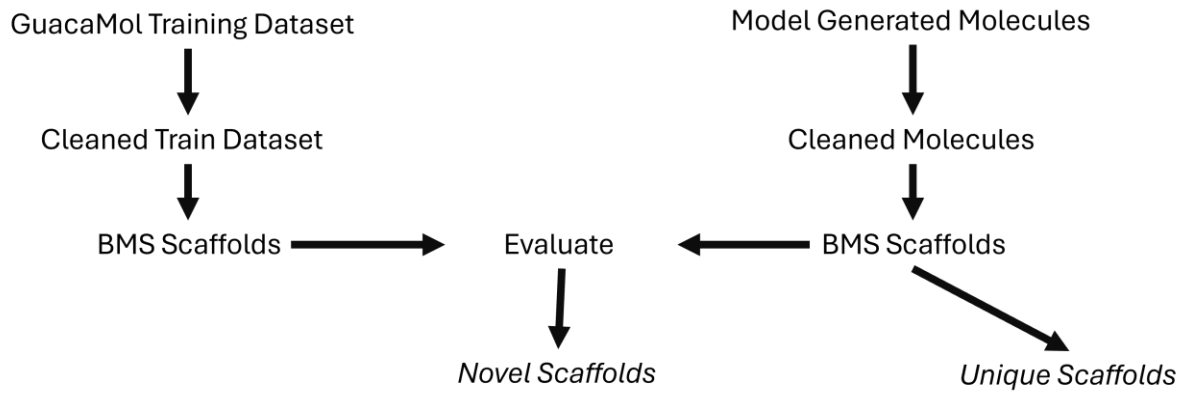


Figure 9-1. The Bemis-Murcko Scaffold Analysis Workflow.

9.2.3 Experimental Setup

The performance of the RNN-LSTM baseline model was compared with data augmentations at the subset-loop stage with 5 loops. Augmentations performed included atom-fusion 3, atom-fusion 3 with FG mask, and graph-fusion 1, as described in the previous chapter. The performance of the models was evaluated using MolScore's 5-HT_{2A} selectivity benchmark during goal-directed generation. Additionally, the QED and SAscores were measured.

To perform the score trajectory analysis, the top-1, top-10, top-100, and the overall average scores were calculated at each epoch. The results at each epoch are from three repeats. This gives both a mean and standard deviation.

To perform the sampled vs augmented analysis, the molecules were labelled during goal-directed generation. The sampled label indicates that the molecule was generated directly from the model, i.e. by sampling the model. The augmented label indicates that the molecule was produced by the augmentation process. All of the SMILES generated during a goal-directed generation run were grouped according to their labels and their scores plotted. The results are from three repeats and shows both means and standard deviations.

9.3 Results and Discussion

9.3.1 5-HT_{2A} Selectivity Benchmark Tasks

Table 9-2 and Table 9-3 show the average scores of the top 300 molecules generated for the 5-HT_{2A} and 5-HT_{2A}-Serotonin-Dopamine tasks, respectively, together with the average QED and SAscores. Results indicate that augmentation with graph-fusion 1 improved the task score for both 5-HT2A (0.951±0.013) and 5-HT2A-Serotonin-Dopamine (0.869±0.024) when compared to the baseline without augmentation. However, this comes at the cost of QED and SAscore. In contrast, augmentation with the atom-fusion 3 was only beneficial for the 5-HT2A-Serotonin-Dopamine task (0.798±0.019). A potential explanation is that in the 5-HT_{2A} task, the baseline without augmentation already generates high scoring molecules and, therefore, it may not be possible to improve on this.

Score Trajectory

Figure 9-2 and Figure 9-3 show the trajectory of the average 5-HT_{2A} and 5-HT_{2A}-Serotonin-Dopamine tasks scores, respectively, for the top-1, top-10, and top-100 molecules at each epoch during the RNN-LSTM goal-directed generation with and without augmentation. Results indicate that, for both tasks, the models with augmentation generated higher scoring molecules at earlier epochs. This suggests that augmentation helped reduce the number of fine-tuning steps, i.e. epochs, required by the model, which can be resource intensive. A limitation of this trajectory analysis is that it does not consider that the augmentation at the subset loop requires extra scoring function calls, which can also be resource intensive, e.g. when performing docking. There is therefore a trade-off between reducing the number of fine-tuning steps or reducing scoring function calls.

Table 9-2. The 5-HT_{2A}, SAscore, and QED Scores of the Molecules Generated by the Baseline RNN-LSTM and Augmented RNN-LSTM with Different Operators at the Subset Loop = 5 Point. The results are averaged across the 300 molecules generated across triplicate repeats. Results in bold indicate significantly improved score compared to the RNN-LSTM baseline. The p-value was obtained using a pairwise comparison t-test against the RNN-LSTM. The full result for the statistical testing is in Appendix Table 0-1. The p-value indicates levels of significance: **** (p <= 0.0001), *** (p <= 0.001), ** (p <= 0.01), * (p <= 0.05), and ns (p > 0.05).

Model	Score	p-value	QED	p-value	SAscore	p-value
RNN-LSTM	0.942±0.013	-	0.564±0.182	-	2.158±0.276	-
Atom-Fusion 3	0.937±0.015	***	0.589±0.173	ns	**2.054±0.244**	****
Atom-Fusion 3 FG Mask	0.934±0.017	****	0.600±0.179	*	**2.008±0.271**	****
Graph Fusion 1	**0.951±0.013**	****	0.541±0.197	ns	2.481±0.656	****

Table 9-3. The 5-HT_{2A}-Serotonin-Dopamine MPO, SAscore, and QED Scores of the Molecules Generated by the Baseline RNN-LSTM and Augmented RNN-LSTM with Different Operators at the Subset Loop = 5 Point. The results are averaged across the 300 molecules generated across triplicate repeats. Results in bold indicate a significantly improved score compared to the RNN-LSTM baseline. The p-value was obtained using a pairwise comparison t-test against the RNN-LSTM. The full result for the statistical testing is in Appendix Table 0-2. The p-value indicates levels of significance: **** (p <= 0.0001), *** (p <= 0.001), ** (p <= 0.01), * (p <= 0.05), and ns (p > 0.05).

Model	MPO	p-value	QED	p-value	SAscore	p-value
RNN-LSTM	0.791±0.016	-	0.529±0.183	-	2.445±0.269	-
Atom-Fusion 3	**0.798±0.019**	***	**0.566±0.191**	*	3.017±0.577	****
Atom-Fusion 3 FG Mask	0.791±0.027	ns	**0.615±0.171**	****	3.063±0.586	****
Graph Fusion 1	**0.869±0.024**	****	0.472±0.202	***	3.270±0.574	****

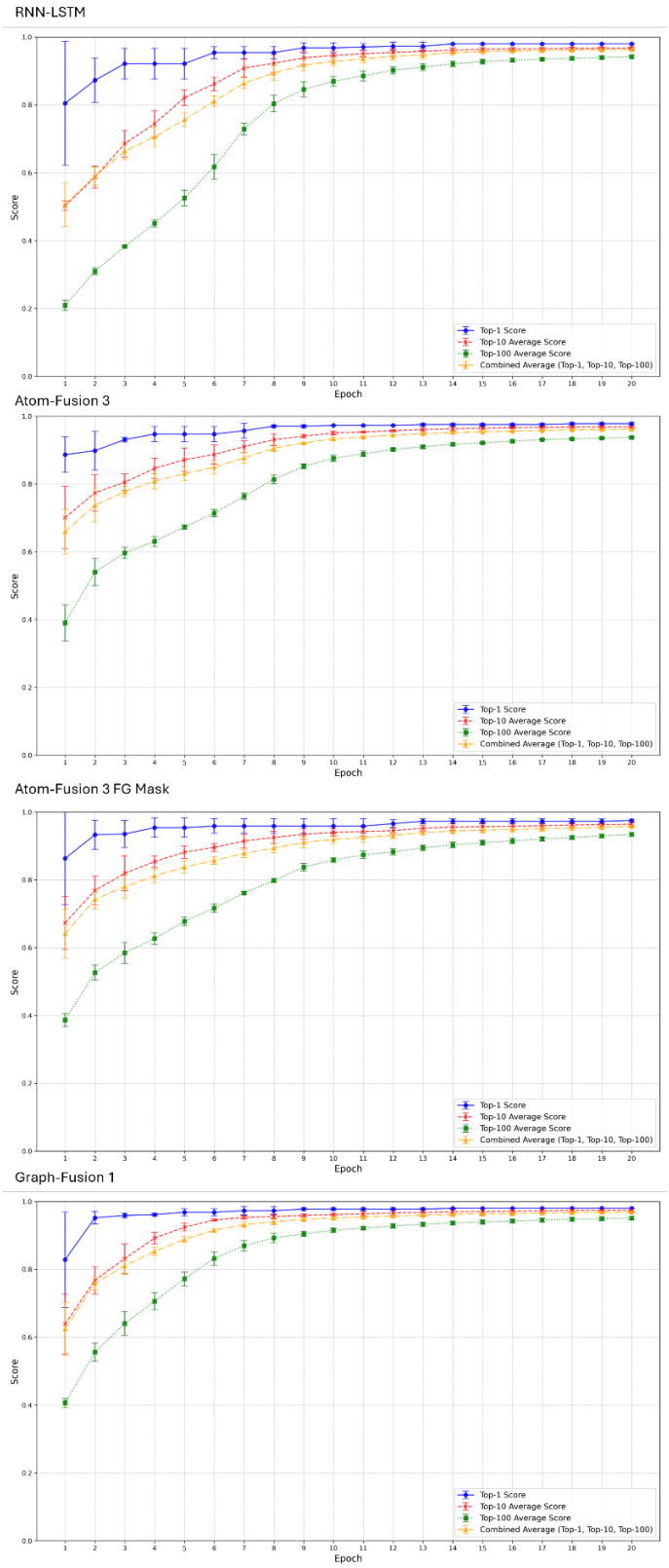


Figure 9-2. The Average 5-HT$_{2A}$ Scores of the Top-k Molecules at Each Epoch During the Optimisation of the Baseline RNN-LSTM and Augmented RNN-LSTM. The augmented RNN-LSTM models were augmented with atom-fusion 3, atom-fusion 3 with functional group masking, and graph-fusion 1 at the subset loop stage with loops = 5. The results shown are averaged across thee repeats with standard deviation.

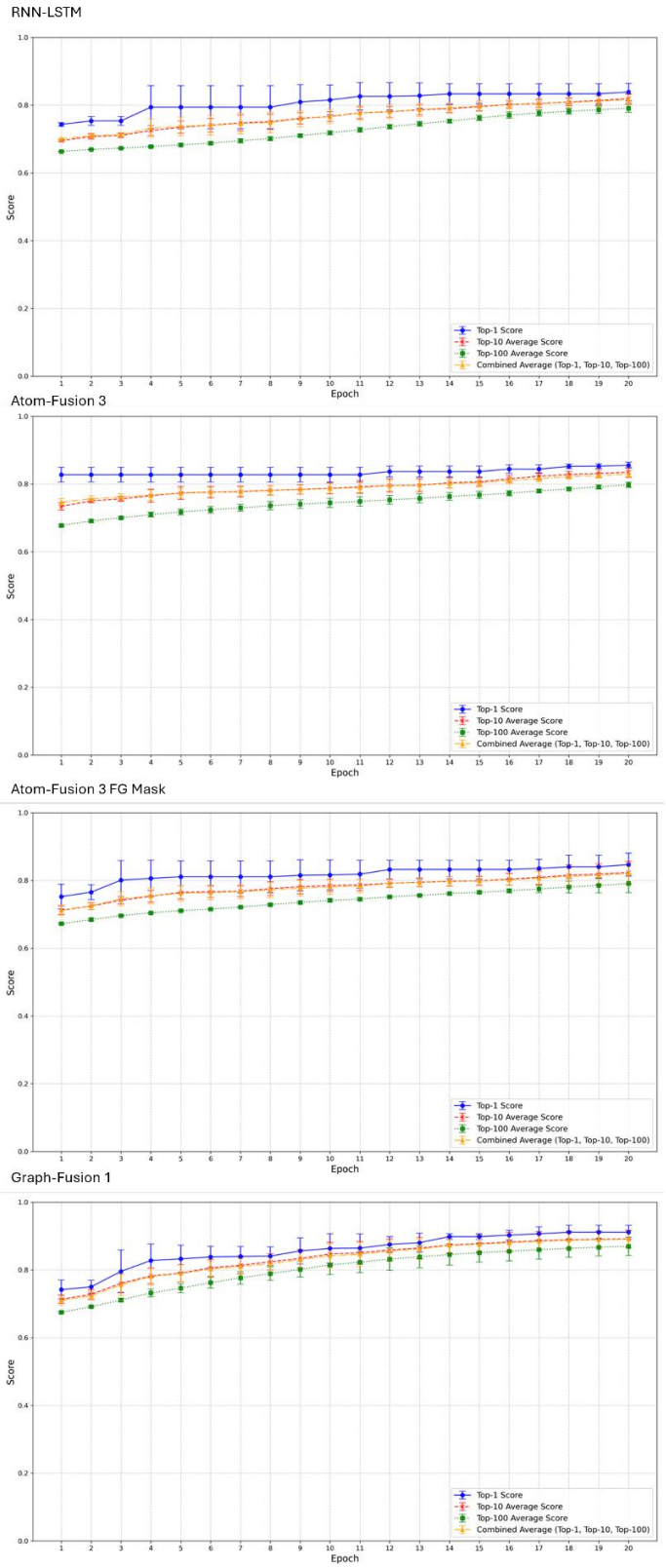


Figure 9-3. The Average 5-HT$_{2A}$-Serotonin-Dopamine MPO Scores of the Top-k Molecules at Each Epoch During the Optimisation of the Baseline RNN-LSTM and Augmented RNN-LSTM. The augmented RNN-LSTM models were augmented with atom-fusion 3, atom-fusion 3 with functional group masking, and graph-fusion 1 at the subset loop stage with loops = 5. The results shown are averaged across thee repeats with standard deviation.

Sampled SMILES vs Augmented SMILES

The next set of analyses is to differentiate SMILES generated by sampling the model from those generated by the augmentation process. The aim here is to identify how each type of generation contributes to the overall MPO scores of the molecules.

Figure 9-4 show the box plots and Table 9-4 show the summary statistics of the 5-HT_{2A} scores from the full SMILES generated by sampling and augmentation. Results generally indicate that the augmented SMILES have lower 5-HT_{2A} scores compared to the sampled SMILES. On the other hand, Figure 9-5 show the box plots and Table 9-5 show the summary statistics of the 5-HT_{2A}-Serotonin-Dopamine scores from the full SMILES generated by sampling and augmentation. Like the results above, augmented SMILES tend to have lower scores compared to the sampled SMILES. Additionally, the spread of the data is much smaller for the 5-HT_{2A}-Serotonin-Dopamine task. This may suggest the model is exploring a narrow chemical space as the scoring function is multi-objective. Therefore, it may be harder for the model to improve upon the 5-HT_{2A}-Serotonin-Dopamine task. This can be seen with the lower max scores across the generated molecules (Table 9-5).

Interestingly, results indicate that in both tasks, graph-fusion 1 augmentation improved the mean task scores of the molecules generated from sampling for both tasks (5-HT_{2A} = 0.500 ± 0.278 and 5-HT_{2A}-Serotonin-Dopamine = 0.713 ± 0.065) compared to the RNN-LSTM without augmentation (5-HT_{2A} = 0.460 ± 0.292 and 5-HT_{2A}-Serotonin-Dopamine = 0.677 ± 0.035). In contrast, both the atom-fusion 3 and atom-fusion 3 FG mask augmentations had an overall negative effect on the mean task scores of the molecules generated from sampling for both tasks (5-HT_{2A}: atom-fusion 3 sampled = 0.420 ± 0.248 and atom-fusion 3 FG mask = 0.391 ± 0.274; 5-HT_{2A}-Serotonin-Dopamine: atom-fusion 3 sampled = 0.673 ± 0.034 and atom-fusion 3 FG mask = 0.673 ± 0.032) when compared to the sampled scores for the baseline.

These results suggest that the augmented SMILES generated by the graph-fusion has a positive effect on the RNN-LSTM's ability to generate high scoring molecules, unlike atom-fusion. This may be because the graph-fusion method incorporates chemical knowledge into its process.

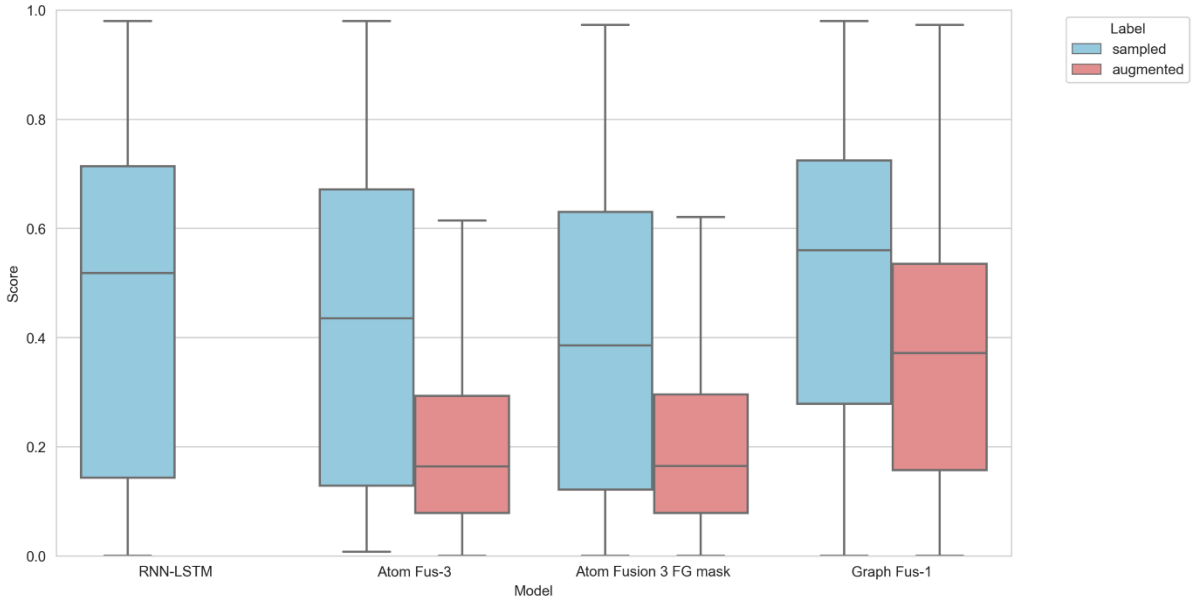


Figure 9-4. The Boxplot of the 5-HT_{2A} Scores from the Sampled vs Augmented Generated SMILES During Goal-Directed Generation. The results shown are calculated from the total SMILES generated by the models across repeats.

Table 9-4. Summary Statistics for Average 5-HT_{2A} Scores of the Sampled vs Augmented Generated SMILES. The results shown are calculated from the total SMILES generated by the models across repeats.

			Objective Score			
Model	**Label**	**Count**	**Mean**	**std**	**min**	**max**
RNN-LSTM	sampled	28145	0.460	0.292	0.000	0.980
Atom-Fusion 3	augmented	119523	0.208	0.167	0.000	0.973
	sampled	27879	0.420	0.284	0.007	0.980
Atom-Fusion 3 FG Mask	augmented	118568	0.207	0.164	0.000	0.973
	sampled	28064	0.391	0.274	0.000	0.973
Graph-Fusion 1	augmented	107718	0.360	0.233	0.000	0.973
	sampled	29042	0.500	0.278	0.000	0.980

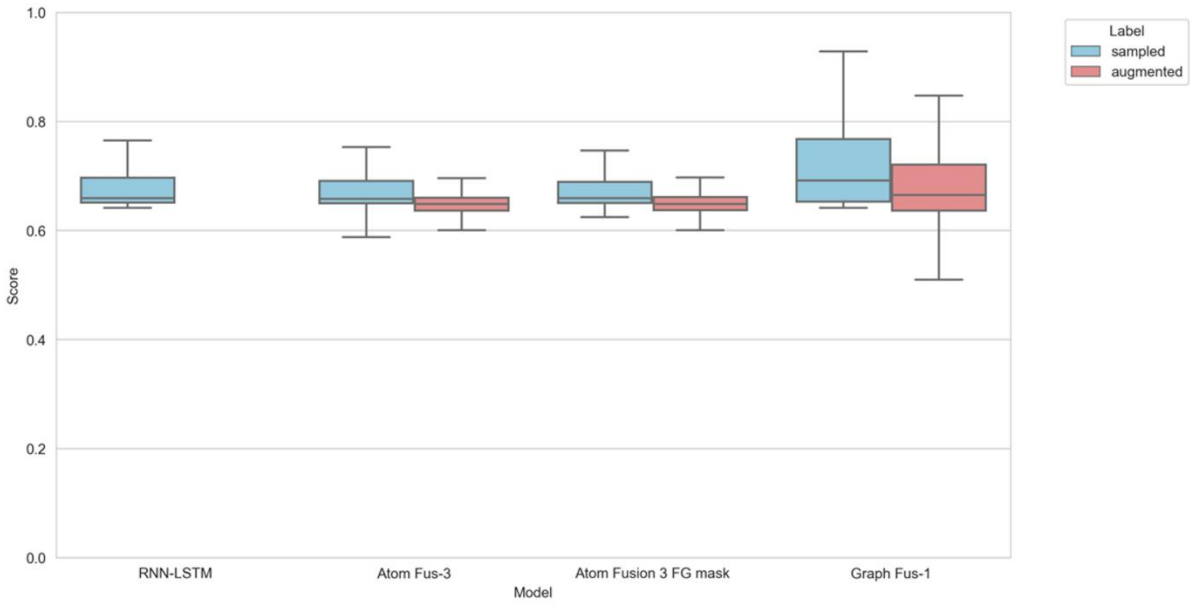


Figure 9-5. The Boxplot of the 5-HT$_{2A}$-Serotonin-Dopamine MPO Scores of the Sampled vs Augmented Generated SMILES During Goal-Directed Generation. The results shown are calculated from the total SMILES generated by the models across repeats.

Table 9-5. Summary Statistics for Average 5-HT$_{2A}$-Serotonin-Dopamine Scores of the Sampled vs Augmented Generated SMILES. The results shown are calculated from the total SMILES generated by the models across repeats.

			Objective Score			
Model	**Label**	**Count**	**Mean**	**std**	**min**	**max**
RNN-LSTM	sampled	29902	0.677	0.035	0.642	0.856
Atom-Fusion 3	augmented	117431	0.633	0.093	0.000	0.830
	sampled	30121	0.673	0.034	0.545	0.863
Atom-Fusion 3 FG Mask	augmented	116349	0.634	0.097	0.000	0.849
	sampled	30052	0.673	0.032	0.624	0.878
Graph-Fusion 1	augmented	110002	0.605	0.227	0.000	0.901
	sampled	30277	0.713	0.065	0.641	0.929

BM Scaffold Analysis – Sampled SMILES

Table 9-6 and Table 9-7 show the BM scaffold analysis for the sampled molecules, i.e. not augmented molecules, for the different models configured for the goal-directed generation of the 5-HT_{2A} and 5-HT_{2A}-Serotonin-Dopamine tasks, respectively. Results indicate that graph-fusion 1 improved the number of novel scaffolds sampled for both tasks when compared to the model without augmentation. This suggests that augmentation with graph-fusion 1 improved the RNN-LSTM's ability to explore the chemical space more widely. In contrast, atom-fusion 3 and atom-fusion 3 with FG mask was only able to improve the number of novel scaffolds generated for the and 5-HT_{2A}-Serotonin-Dopamine task.

Table 9-6. Bemis-Murcko Scaffold Analysis of the Sampled SMILES for the 5-HT_{2A} Task for the Baseline and Augmented RNN-LSTM Models. Generated molecules are the total SMILES produced across three repeats with duplicates removed. Unique scaffolds indicate the number of distinct scaffolds in the generated molecules. Novel scaffolds indicate the number of unique scaffolds not found in the GuacaMol dataset.

Model	Generated molecules	Unique Scaffolds	Novel Scaffolds
RNN-LSTM	28145	20910	17804
Atom-Fusion 3	27879	19097	15890
Atom-Fusion 3 FG Mask	28064	18825	15634
Graph-Fusion 1	29042	23540	21380

Table 9-7. Bemis-Murcko Scaffold Analysis of the Sampled SMILES for the 5-HT_{2A}-Serotonin-Dopamine Task for the Baseline and Augmented RNN-LSTM Models. Generated molecules are SMILES produced across three repeats with duplicates removed. Unique scaffolds indicate the number of distinct scaffolds in the generated molecules. Novel scaffolds indicate the number of unique scaffolds not found in the GuacaMol dataset.

Model	Generated molecules	Unique Scaffolds	Novel Scaffolds
RNN-LSTM	29902	27847	23062
Atom-Fusion 3	30121	28650	25084
Atom-Fusion 3 FG Mask	30052	28768	25258
Graph-Fusion 1	30277	28821	26434

9.4 Conclusion

This chapter applied the augmentation operators developed for generative *de novo* design models in the previous chapters to more realistic drug discovery problems, based on QSAR models, while also extending the analysis of the output to include scaffold analysis. Results showed that graph-fusion 1 is the most promising operator, as it improved model performance during goal-directed generation both in terms of QSAR scores and the number of novel BM scaffolds generated. However, the main limitation of the graph-fusion operator is that it has a negative impact on both QED and SAscore. This may make it difficult to synthesise and experimentally validate the molecules generated by this method. Additionally, the atom-fusion augmentation did not offer a clear benefit, as seen in the previous chapters. This may indicate that the operator struggles in more realistic evaluation as it does not incorporate chemical knowledge during its process, in contrast to graph-fusion. Additionally future work should perform the sampled vs augmented and BM scaffold analyses on the top-300 molecules. This is important as high scoring molecules tend to be considered as leads in real-world drug discovery. Overall, this chapter shows that performing augmentation on generative *de novo* design models is beneficial. However, this must be constrained with chemical knowledge.

Chapter 10. Conclusions

10.1 Conclusions

The thesis has described the development of novel data augmentation operators for generative *de novo* design. The performance of generative *de novo* design models is linked to the size of the training dataset they have been trained on. The data augmentation operators developed throughout the thesis have been shown to improve the performance of generative *de novo* design models on goal-directed generation tasks.

Chapter 6 reported on experiments designed to investigate the effect of performing data augmentation on the primary dataset used to pretrain a generative de novo design model before being tasked on goal-directed generation. The types of data augmentations performed included duplication, randomised SMILES, and BRICS enumeration. The full suite of the GuacaMol benchmark was used to evaluate goal-directed generation performance of the baseline and augmented models. Results indicated that performing any type of data augmentation generally improved model performance across all of GuacaMol's tasks. An exception, however, was the Sitagliptin MPO, where model performance was not improved by data augmentation. In some cases, data augmentation had a negative effect on the model's performance. This indicated a potential need for alternative forms of data augmentation. Additionally, it was hypothesised that performing a more direct data augmentation on the secondary, finetuning dataset may help improve model performance in goal-directed generation tasks.

Chapter 7 described the development of the NLP-inspired operators, which includes atom-delete, atom-insert, atom-swap, and atom-fusion. The first step was to refine the operators by adding constraints in the algorithm to ensure that the augmented SMILES were valid. Next, the best point to introduce the data augmentation operators during the RNN-LSTM's goal-directed generation process was investigated. Results indicate that performing augmentations at the subset loop stage improved model performance the most. Finally, a systematic search was performed to identify appropriate parameters for the different atom-level operators at the subset loop stage.

Chapter 8 further developed the atom-level operators from Chapter 7. The atom-level operators were found to be beneficial for optimising the MPO scores of the generated molecules. However, this had a negative impact on the synthesisability. Therefore, the aim of this chapter was to refine the atom-level operators. The first step was to introduce further constraints on the algorithms by masking atoms that are potentially linked to synthesisability. These masks include the SAscore- and functional group-based masking. Alternative operators from the literature were also identified and investigated. This included bioisosteric substitution and the graph-based operators. Results indicate that the impact of the operators on the synthesisability of the generated molecules may be task dependent. Finally, filtering of the finetuning dataset was also explored. The filters generally had a positive effect on the synthesisability of the generated molecules across the different tasks compared to no filters. However, they had a negative impact on the MPO scores of the generated molecules.

Chapter 9 investigated the data augmentation operators on a set of more realistic drug discovery tasks. MolScore was used to evaluate the models as it provided QSAR models. The tasks used include the 5-HT_{2A} and the 5-HT_{2A}-Dopamaine-Serotonin. Results indicate the graph-fusion operator significantly improved model performance on the objective scores of these more realistic tasks compared to the baseline RNN-LSTM without augmentation. Additionally, it was identified that the graph-fusion increased the number of novel scaffolds sampled from the models.

To conclude, the thesis has shown the benefits of data augmentation on improving generative *de novo* design performance on goal-directed generation performance, in both similarity-based and QSAR-based evaluations. Although this comes at the cost of synthesisability, potential methods have been introduced in this work to address this, including bioisosteric substitution and finetuning dataset filtering. Additionally, it has been identified that the ideal data augmentation operator is likely to be task dependent, similar to the findings of Brinkmann et al. (2025) for distribution learning. Finally, the operators described in this chapter is also expected to be a beneficial alternative for the data augmentation of reinforcement learning-based optimisation (Bjerrum *et al.*, 2023; Guo and Schwaller, 2024).

10.2 Limitations and Future Work

Several limitations of the work described in this thesis and potential future work that may be done to address these are discussed in this section. The main limitation is that performing augmentation at the subset loop stage increases the number of scoring function calls during the goal-directed generation process. On average, 10,000 scoring function calls are performed during a run of a baseline RNN-LSTM. In contrast, data augmentation at the subset loop with l = 5 increases the average number of calls to 40,000. This can therefore be time and resource-intensive. Although this was not a major issue for the scoring functions investigated here it could be for more time-intensive scoring functions such as, for example, docking. A potential solution in these scenarios might be to use active learning. This is a method where an algorithm selects the most informative samples from an unlabelled dataset to label (Dodds *et al.*, 2024). Active learning may therefore reduce the number of scoring function calls used when augmenting at the subset loop stage. Secondly, a more extensive case study to evaluate the data augmentation operators in more realistic drug design settings was not performed due to time constraints. Such scenarios might include docking, referred to above, which provides a prediction of a molecule's binding affinity, i.e., how well the ligand molecule fits into the active site of a target's active site. Also, another potential evaluation metric, which is used in realistic drug scenario, is the virtual ligand efficiency (VLE). VLE penalises models that trivially generate large molecules to improve binding

affinity but at the cost of drug-likeness (Kuntz *et al.*, 1999). Finally, an experiment in the extremely low regime, where the primary pretraining dataset only contains fewer than 5000 molecules, is a potential way to further extend the investigation on the effect of augmenting the primary training dataset on goal-directed generation tasks. Performing data augmentation in this regime has been found to be beneficial distribution learning (Brinkmann *et al.*, 2025).

Appendix A NLP-inspired Operators

RNN-LSTM- Augmentation Points

Table 0-1. The Sitagliptin MPO scores of the molecules generated by the Baselines (RNN-LSTM and Graph-GA) and the RNN-LSTM augmented at different points of the finetuning stage. The results are averaged across the 300 molecules generated across triplicate repeats. Results in bold indicate an improved score compared to the RNN-LSTM baseline.

Baselines	MPO
RNN-LSTM	0.452±0.033
Graph-GA	0.763±0.046
Sample Stage Augmentation	MPO
Atom-Delete	**0.476±0.038**
Atom-Insert	**0.482±0.038**
Atom-Swap	**0.477±0.034**
Atom-Fusion	**0.467±0.038**
Subset Stage Augmentation	MPO
Atom-Delete	**0.473±0.036**
Atom-Insert	**0.468±0.036**
Atom-Swap	**0.454±0.040**
Atom-Fusion	**0.459±0.037**
Randomised N2	0.433±0.043
Randomised N5	**0.456±0.037**
Randomised N10	**0.500±0.040**
Subset-loop l = 5 Stage Augmentation	MPO
Atom-Delete	**0.557±0.038**
Atom-Insert	**0.601±0.026**
Atom-Swap	**0.654±0.028**
Atom-Fusion	**0.659±0.039**

Table 0-2. The Ranolazine MPO scores of the molecules generated by the Baselines (RNN-LSTM and Graph-GA) and the RNN-LSTM augmented at different points of the finetuning stage. The results are averaged across the 300 molecules generated across triplicate repeats. Results in bold indicate an improved score compared to the RNN-LSTM baseline.

Baselines	MPO
RNN-LSTM	0.814±0.013
Graph-GA	0.888±0.009
Sample Stage Augmentation	MPO
Atom-Delete	**0.843±0.008**
Atom-Insert	**0.837±0.008**
Atom-Swap	0.808±0.008
Atom-Fusion	0.811±0.008
Subset Stage Augmentation	MPO
Atom-Delete	**0.824±0.007**
Atom-Insert	**0.824±0.010**
Atom-Swap	0.809±0.008
Atom-Fusion	0.805±0.008
Randomised N2	0.803±0.011
Randomised N5	0.812±0.008
Randomised N10	**0.833±0.008**
Subset-loop *l* = 5 Stage Augmentation	MPO
Atom-Delete	**0.859±0.013**
Atom-Insert	**0.873±0.007**
Atom-Swap	**0.867±0.024**
Atom-Fusion	**0.840±0.004**

Appendix B Chemically-sensible Operators

SAscore and QED Analysis

Table 0-1. The Sitagliptin MPO, SAscore, and QED Scores of the Molecules Generated by the Baselines (RNN-LSTM and Graph-GA) and the RNN-LSTM Augmented with Atom-Fusion at the subset loop = 5 point. The results are averaged across the 300 molecules generated across triplicate repeats. Results in bold indicate a significantly improved score compared to the baselines. The p-value was obtained using a pairwise comparison t-test against the RNN-LSTM (pv RNN) and Graph-GA (pv GA). The full result for the statistical testing is in Appendix Table 0-2. The p-value indicates levels of significance: **** (p <= 0.0001), *** (p <= 0.001), ** (p <= 0.01), * (p <= 0.05), and ns (p > 0.05).

Model	MPO	pv RNN	pv GA	QED	pv RNN	pv GA	SAscore	pv RNN	pv GA
RNN-LSTM	0.452±0.033	-	****	0.709±0.124	-	****	2.780±0.577	-	****
Graph-GA	**0.763±0.046**	****	-	0.506±0.142	****	-	5.626±0.380	****	-
Atom Fus-1	**0.687±0.019**	****	****	0.457±0.145	****	****	4.949±0.560	****	****
Atom Fus-2	**0.708±0.047**	**	****	0.670±0.174	****	****	4.848±0.495	****	****
Atom Fus-3	**0.645±0.037**	*	****	0.681±0.176	****	****	5.160±0.535	****	****

Table 0-2. Welch's t-test Results for the Pairwise Comparison of the baselines (RNN-LSTM and Graph-GA) and the augmented RNN-LSTM (with Atom-Fusion) when tasked on the Sitagliptin MPO. The significance indicates: **** (p <= 0.0001), *** (p <= 0.001), ** (p <= 0.01), * (p <= 0.05), and ns (p > 0.05).

Comparison	Score	t-statistic	p-value	Significance
RNN-LSTM vs Graph-GA	MPO	-87.1705	3.14E-299	****
RNN-LSTM vs Graph-GA	QED	16.67923	5.88E-49	****
RNN-LSTM vs Graph-GA	SAscore	-60.539	6.20E-170	****
RNN-LSTM vs Atom Fus-1	MPO	-90.3114	1.23E-208	****
RNN-LSTM vs Atom Fus-1	QED	20.64292	3.30E-67	****
RNN-LSTM vs Atom Fus-1	SAscore	-41.1742	4.95E-146	****
RNN-LSTM vs Atom Fus-2	MPO	-71.416	2.31E-260	****
RNN-LSTM vs Atom Fus-2	QED	2.876078	0.004203	**
RNN-LSTM vs Atom Fus-2	SAscore	-40.9827	1.72E-138	****
RNN-LSTM vs Atom Fus-3	MPO	-61.0961	9.87E-218	****
RNN-LSTM vs Atom Fus-3	QED	2.038138	0.042076	*
RNN-LSTM vs Atom Fus-3	SAscore	-45.9559	8.92E-159	****
Graph-GA vs Atom Fus-1	MPO	26.02012	4.65E-88	****
Graph-GA vs Atom Fus-1	QED	4.247546	2.51E-05	****
Graph-GA vs Atom Fus-1	SAscore	17.32738	1.49E-53	****
Graph-GA vs Atom Fus-2	MPO	14.48126	5.88E-41	****
Graph-GA vs Atom Fus-2	QED	-12.6199	2.08E-32	****
Graph-GA vs Atom Fus-2	SAscore	21.59403	1.02E-75	****
Graph-GA vs Atom Fus-3	MPO	34.39675	5.93E-141	****
Graph-GA vs Atom Fus-3	QED	-13.344	1.47E-35	****
Graph-GA vs Atom Fus-3	SAscore	12.30858	7.33E-31	****
Atom Fus-1 vs Atom Fus-2	MPO	-6.99938	1.10E-11	****
Atom Fus-1 vs Atom Fus-2	QED	-16.3598	9.93E-50	****
Atom Fus-1 vs Atom Fus-2	SAscore	2.340865	0.019572	*
Atom Fus-1 vs Atom Fus-3	MPO	17.51161	7.96E-53	****
Atom Fus-1 vs Atom Fus-3	QED	-17.0462	4.74E-53	****
Atom Fus-1 vs Atom Fus-3	SAscore	-4.71946	2.95E-06	****
Atom Fus-2 vs Atom Fus-3	MPO	18.17993	1.65E-58	****
Atom Fus-2 vs Atom Fus-3	QED	-0.76985	0.441694	ns
Atom Fus-2 vs Atom Fus-3	SAscore	-7.41653	4.17E-13	****

Table 0-3. The Zaleplon MPO, SAscore, and QED Scores of the Molecules Generated by the Baselines (RNN-LSTM and Graph-GA) and the RNN-LSTM Augmented with Atom-Fusion at the subset loop = 5 point. The results are averaged across the 300 molecules generated across triplicate repeats. Results in bold indicate an improved and significant score compared to the RNN-LSTM baseline. The p-value was obtained using a pairwise comparison t-test against the RNN-LSTM (pv RNN) and Graph-GA (pv GA). The full result for the statistical testing is in Appendix Table 0-4. The p-value indicates levels of significance: **** (p <= 0.0001), *** (p <= 0.001), ** (p <= 0.01), * (p <= 0.05), and ns (p > 0.05).

Model	MPO	pv RNN	pv GA	QED	pv RNN	pv GA	SAscore	pv RNN	pv GA
RNN-LSTM	0.562±0.019	-	****	0.688±0.092	-	****	2.323±0.179	-	****
Graph-GA	**0.681±0.021**	****	-	0.561±0.110	****	-	2.774±0.248	****	-
Atom Fus-1	**0.680±0.027**	****	ns	0.573±0.103	****	ns	3.042±0.291	****	****
Atom Fus-2	**0.665±0.022**	****	****	0.534±0.101	****	****	2.855±0.125	****	**
Atom Fus-3	**0.650±0.006**	****	****	0.569±0.057	****	ns	2.811±0.129	****	ns

Table 0-4. Welch's t-test Results for the Pairwise Comparison of the baselines (RNN-LSTM and Graph-GA) and the augmented RNN-LSTM (with Atom-Fusion) when tasked on the Zaleplon MPO. The significance indicates: **** (p <= 0.0001), *** (p <= 0.001), ** (p <= 0.01), * (p <= 0.05), and ns (p > 0.05).

Comparison	Score	t-statistic	p-value	Significance
RNN-LSTM vs Graph-GA	MPO	-65.3377	1.16E-229	****
RNN-LSTM vs Graph-GA	QED	13.62829	2.27E-35	****
RNN-LSTM vs Graph-GA	SAscore	-22.4982	2.63E-72	****
RNN-LSTM vs Atom Fus-1	MPO	-60.3286	1.53E-239	****
RNN-LSTM vs Atom Fus-1	QED	14.05521	1.20E-38	****
RNN-LSTM vs Atom Fus-1	SAscore	-35.741	2.20E-140	****
RNN-LSTM vs Atom Fus-2	MPO	-56.1121	1.44E-209	****
RNN-LSTM vs Atom Fus-2	QED	17.69005	2.93E-54	****
RNN-LSTM vs Atom Fus-2	SAscore	-38.772	2.31E-148	****
RNN-LSTM vs Atom Fus-3	MPO	-71.3467	2.98E-205	****
RNN-LSTM vs Atom Fus-3	QED	17.19017	4.01E-51	****
RNN-LSTM vs Atom Fus-3	SAscore	-34.2293	6.59E-129	****
Graph-GA vs Atom Fus-1	MPO	0.264323	0.791637	ns
Graph-GA vs Atom Fus-1	QED	-1.24047	0.215446	ns
Graph-GA vs Atom Fus-1	SAscore	-11.2758	1.86E-26	****
Graph-GA vs Atom Fus-2	MPO	8.013182	9.54E-15	****
Graph-GA vs Atom Fus-2	QED	2.682443	0.007582	**
Graph-GA vs Atom Fus-2	SAscore	-4.32743	2.02E-05	****
Graph-GA vs Atom Fus-3	MPO	20.5756	1.18E-56	****
Graph-GA vs Atom Fus-3	QED	-0.94434	0.345681	ns
Graph-GA vs Atom Fus-3	SAscore	-1.91441	0.056425	ns
Atom Fus-1 vs Atom Fus-2	MPO	7.261756	1.36E-12	****
Atom Fus-1 vs Atom Fus-2	QED	4.329291	1.80E-05	****
Atom Fus-1 vs Atom Fus-2	SAscore	10.00946	2.53E-21	****
Atom Fus-1 vs Atom Fus-3	MPO	18.09179	5.81E-52	****
Atom Fus-1 vs Atom Fus-3	QED	0.536512	0.59185	ns
Atom Fus-1 vs Atom Fus-3	SAscore	12.11998	2.17E-29	****
Atom Fus-2 vs Atom Fus-3	MPO	9.608428	4.26E-19	****
Atom Fus-2 vs Atom Fus-3	QED	-4.48978	9.46E-06	****
Atom Fus-2 vs Atom Fus-3	SAscore	3.638284	0.000308	***

Substructure Masking

Atom-Delete

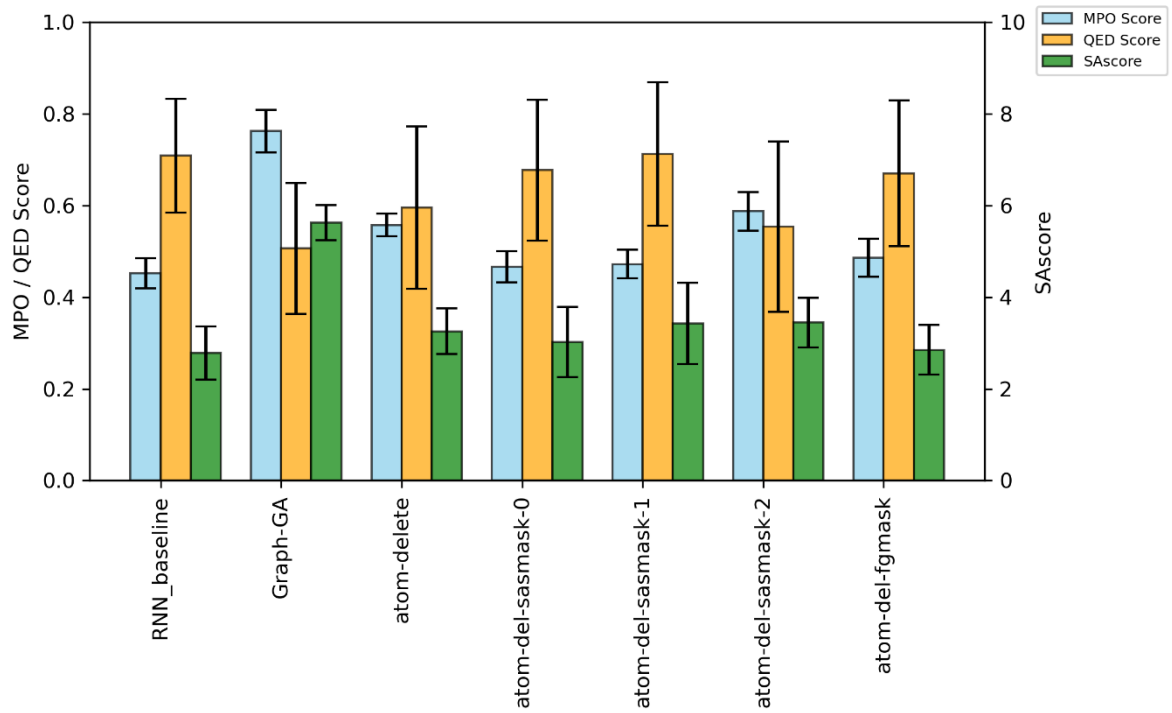


Figure 0-1. The Average Sitagliptin MPO, SAscore, and QED Scores of the Molecules Generated by the RNN-LSTM Atom-Delete Augmented Model with and without Masking. The atom-delete with SAscore mask also included thresholds of 0, 1, and 2. The results shown are averaged across three repeats with standard deviation. The left y-axis represents the scale of values for the MPO and QED scores. The right y-axis represents the scale of values for the SAscores.

Table 0-5. Welch's t-test Results for the Pairwise Comparison of the augmented RNN-LSTM using Atom-Delete with and without Masking when tasked on the Sitagliptin MPO. The significance indicates: **** (p <= 0.0001), *** (p <= 0.001), ** (p <= 0.01), * (p <= 0.05), and ns (p > 0.05).

Comparison	Score	t-statistic	p-value	Significance
Atom Delete-015 vs Atom Delete-015 SAS mask 0	MPO	27.20935	5.66E-92	****
Atom Delete-015 vs Atom Delete-015 SAS mask 0	QED	-4.04525	6.16E-05	****
Atom Delete-015 vs Atom Delete-015 SAS mask 0	SAscore	8.037434	1.62E-14	****
Atom Delete-015 vs Atom Delete-015 SAS mask 1	MPO	28.7179	1.57E-105	****
Atom Delete-015 vs Atom Delete-015 SAS mask 1	QED	-2.4333	0.015335	*
Atom Delete-015 vs Atom Delete-015 SAS mask 1	SAscore	-2.57463	0.010452	*
Atom Delete-015 vs Atom Delete-015 SAS mask 2	MPO	-11.8109	1.55E-28	****
Atom Delete-015 vs Atom Delete-015 SAS mask 2	QED	2.773935	0.00575	**
Atom Delete-015 vs Atom Delete-015 SAS mask 2	SAscore	-2.21634	0.02714	*
Atom Delete-015 vs Atom Delete-015 FG mask	MPO	19.08729	2.35E-57	****
Atom Delete-015 vs Atom Delete-015 FG mask	QED	-1.98544	0.047696	*
Atom Delete-015 vs Atom Delete-015 FG mask	SAscore	11.34691	4.76E-26	****
Atom Delete-015 SAS mask 0 vs Atom Delete-015 SAS mask 1	MPO	-3.30968	0.001023	**
Atom Delete-015 SAS mask 0 vs Atom Delete-015 SAS mask 1	QED	1.397351	0.163034	ns
Atom Delete-015 SAS mask 0 vs Atom Delete-015 SAS mask 1	SAscore	-8.37652	7.79E-16	****
Atom Delete-015 SAS mask 0 vs Atom Delete-015 SAS mask 2	MPO	-36.4503	1.39E-130	****
Atom Delete-015 SAS mask 0 vs Atom Delete-015 SAS mask 2	QED	7.226681	2.12E-12	****
Atom Delete-015 SAS mask 0 vs Atom Delete-015 SAS mask 2	SAscore	-9.16388	2.88E-18	****
Atom Delete-015 SAS mask 0 vs Atom Delete-015 FG mask	MPO	-5.86491	9.22E-09	****
Atom Delete-015 SAS mask 0 vs Atom Delete-015 FG mask	QED	2.10798	0.035632	*
Atom Delete-015 SAS mask 0 vs Atom Delete-015 FG mask	SAscore	1.443112	0.149789	ns
Atom Delete-015 SAS mask 1 vs Atom Delete-015 SAS mask 2	MPO	-39.4604	1.02E-152	****
Atom Delete-015 SAS mask 1 vs Atom Delete-015 SAS mask 2	QED	5.270877	2.09E-07	****
Atom Delete-015 SAS mask 1 vs Atom Delete-015 SAS mask 2	SAscore	0.867616	0.386125	ns
Atom Delete-015 SAS mask 1 vs Atom Delete-015 FG mask	MPO	-3.49745	0.000527	***
Atom Delete-015 SAS mask 1 vs Atom Delete-015 FG mask	QED	0.574225	0.566112	ns
Atom Delete-015 SAS mask 1 vs Atom Delete-015 FG mask	SAscore	10.53136	4.42E-23	****
Atom Delete-015 SAS mask 2 vs Atom Delete-015 FG mask	MPO	28.07504	1.65E-96	****
Atom Delete-015 SAS mask 2 vs Atom Delete-015 FG mask	QED	-4.98403	8.85E-07	****
Atom Delete-015 SAS mask 2 vs Atom Delete-015 FG mask	SAscore	12.2413	6.00E-30	****

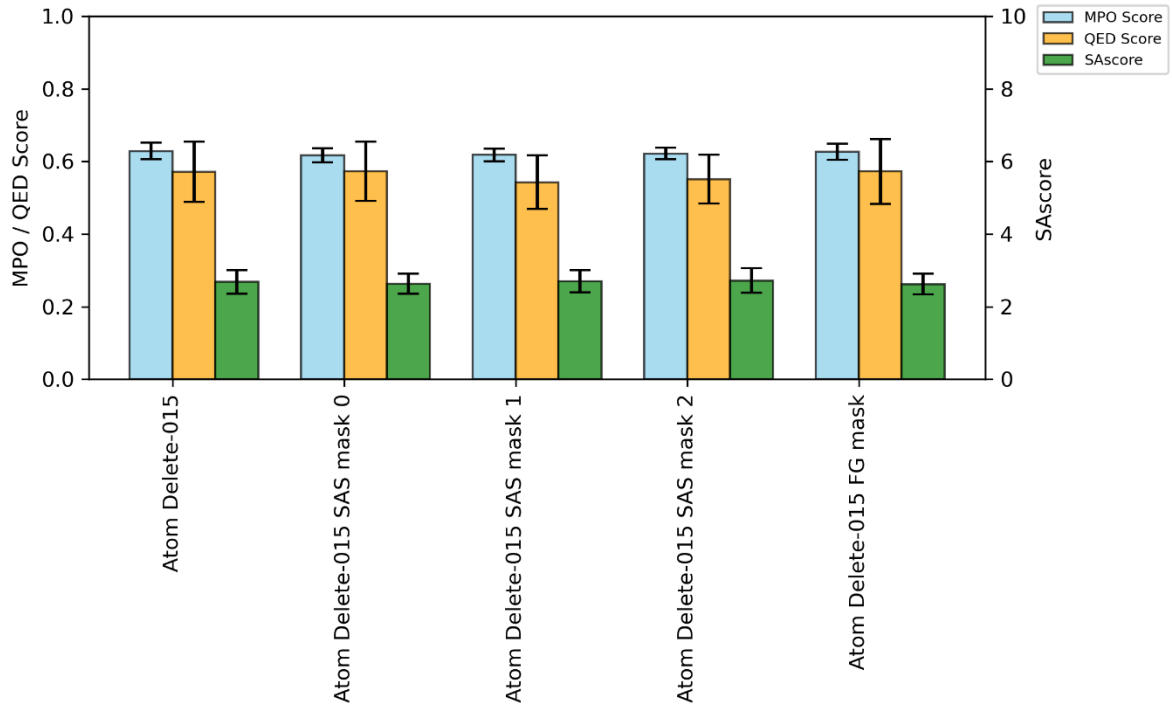


Figure 0-2. The Average Zaleplon MPO, SAscore, and QED Scores of the Molecules Generated by the RNN-LSTM Atom-Delete Augmented Model with and without Masking. The atom-delete with SAscore mask also included thresholds of 0, 1, and 2. The results shown are averaged across three repeats with standard deviation. The left y-axis represents the scale of values for the MPO and QED scores. The right y-axis represents the scale of values for the SAscores.

Table 0-6. Welch's t-test Results for the Pairwise Comparison of the augmented RNN-LSTM using Atom-Delete with and without Masking when tasked on the Zaleplon MPO. The significance indicates: **** (p <= 0.0001), *** (p <= 0.001), ** (p <= 0.01), * (p <= 0.05), and ns (p > 0.05).

Comparison	Score	t-statistic	p-value	Significance
Atom Delete-015 vs Atom Delete-015 SAS mask 0	MPO	6.095319	2.21E-09	****
Atom Delete-015 vs Atom Delete-015 SAS mask 0	QED	-0.17597	0.860389	ns
Atom Delete-015 vs Atom Delete-015 SAS mask 0	SAscore	1.883302	0.06025	ns
Atom Delete-015 vs Atom Delete-015 SAS mask 1	MPO	5.780892	1.36E-08	****
Atom Delete-015 vs Atom Delete-015 SAS mask 1	QED	4.004394	7.22E-05	****
Atom Delete-015 vs Atom Delete-015 SAS mask 1	SAscore	-0.67004	0.503163	ns
Atom Delete-015 vs Atom Delete-015 SAS mask 2	MPO	3.716268	0.000227	***
Atom Delete-015 vs Atom Delete-015 SAS mask 2	QED	2.975449	0.003072	**
Atom Delete-015 vs Atom Delete-015 SAS mask 2	SAscore	-1.21568	0.22472	ns
Atom Delete-015 vs Atom Delete-015 FG mask	MPO	0.95498	0.340032	ns
Atom Delete-015 vs Atom Delete-015 FG mask	QED	-0.10453	0.916785	ns
Atom Delete-015 vs Atom Delete-015 FG mask	SAscore	2.299166	0.021901	*
Atom Delete-015 SAS mask 0 vs Atom Delete-015 SAS mask 1	MPO	-0.55926	0.576262	ns
Atom Delete-015 SAS mask 0 vs Atom Delete-015 SAS mask 1	QED	4.107708	4.75E-05	****
Atom Delete-015 SAS mask 0 vs Atom Delete-015 SAS mask 1	SAscore	-2.57008	0.0105	*
Atom Delete-015 SAS mask 0 vs Atom Delete-015 SAS mask 2	MPO	-2.94494	0.003397	**
Atom Delete-015 SAS mask 0 vs Atom Delete-015 SAS mask 2	QED	3.101267	0.002049	**
Atom Delete-015 SAS mask 0 vs Atom Delete-015 SAS mask 2	SAscore	-3.03426	0.002556	**
Atom Delete-015 SAS mask 0 vs Atom Delete-015 FG mask	MPO	-5.22857	2.53E-07	****
Atom Delete-015 SAS mask 0 vs Atom Delete-015 FG mask	QED	0.066418	0.947072	ns
Atom Delete-015 SAS mask 0 vs Atom Delete-015 FG mask	SAscore	0.396885	0.691626	ns
Atom Delete-015 SAS mask 1 vs Atom Delete-015 SAS mask 2	MPO	-2.47673	0.013638	*
Atom Delete-015 SAS mask 1 vs Atom Delete-015 SAS mask 2	QED	-1.27218	0.203996	ns
Atom Delete-015 SAS mask 1 vs Atom Delete-015 SAS mask 2	SAscore	-0.57493	0.565638	ns
Atom Delete-015 SAS mask 1 vs Atom Delete-015 FG mask	MPO	-4.88053	1.45E-06	****
Atom Delete-015 SAS mask 1 vs Atom Delete-015 FG mask	QED	-3.94793	9.08E-05	****
Atom Delete-015 SAS mask 1 vs Atom Delete-015 FG mask	SAscore	2.996452	0.002885	**
Atom Delete-015 SAS mask 2 vs Atom Delete-015 FG mask	MPO	-2.74098	0.006356	**
Atom Delete-015 SAS mask 2 vs Atom Delete-015 FG mask	QED	-2.95267	0.003306	**
Atom Delete-015 SAS mask 2 vs Atom Delete-015 FG mask	SAscore	3.442144	0.000633	***

Atom-Swap

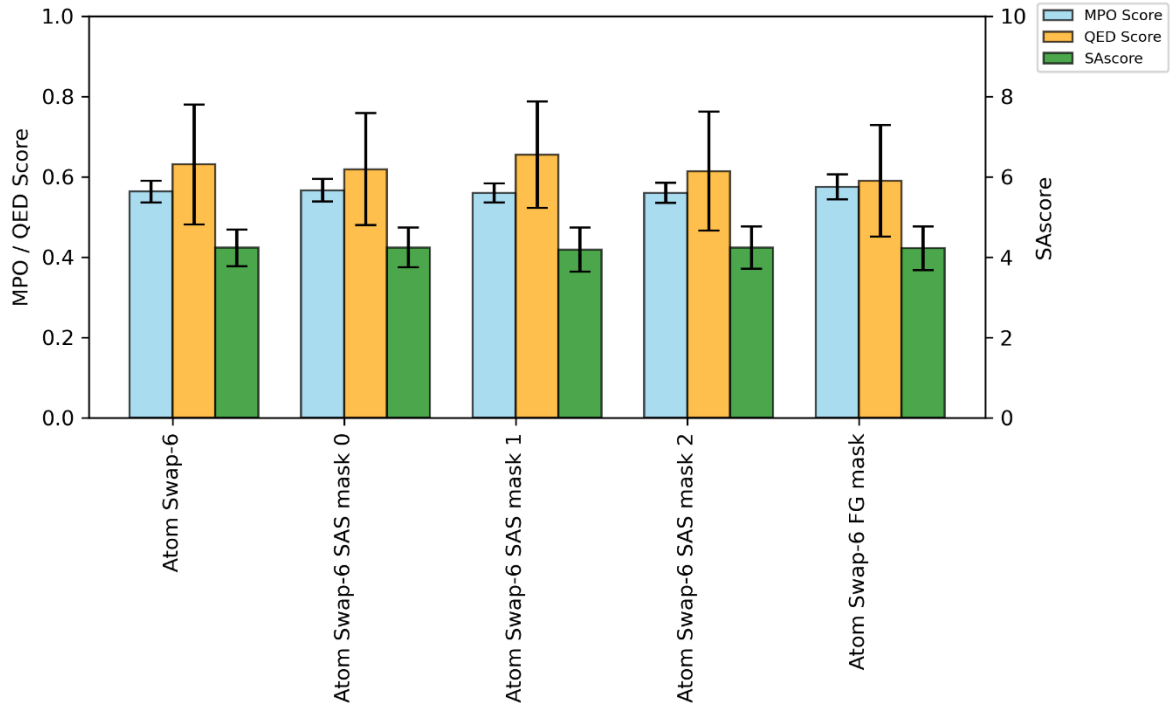


Figure 0-3. The Average Sitagliptin MPO, SAscore, and QED Scores of the Molecules Generated by the RNN-LSTM Atom-Swap Augmented Model with and without Masking. The atom-swap with SAscore mask also included thresholds of 0, 1, and 2. The results shown are averaged across three repeats with standard deviation. The left y-axis represents the scale of values for the MPO and QED scores. The right y-axis represents the scale of values for the SAscores.

Table 0-7. Welch's t-test Results for the Pairwise Comparison of the augmented RNN-LSTM using Atom-Swap with and without Masking when tasked on the Sitagliptin MPO. The significance indicates: **** (p <= 0.0001), *** (p <= 0.001), ** (p <= 0.01), * (p <= 0.05), and ns (p > 0.05).

Comparison	Score	t-statistic	p-value	Significance
Atom Swap-6 vs Atom Swap-6 SAS mask 0	MPO	-1.53278	0.125937	ns
Atom Swap-6 vs Atom Swap-6 SAS mask 0	QED	0.897381	0.369941	ns
Atom Swap-6 vs Atom Swap-6 SAS mask 0	SAscore	-0.20911	0.83444	ns
Atom Swap-6 vs Atom Swap-6 SAS mask 1	MPO	1.294197	0.196205	ns
Atom Swap-6 vs Atom Swap-6 SAS mask 1	QED	-1.94815	0.05196	ns
Atom Swap-6 vs Atom Swap-6 SAS mask 1	SAscore	1.089428	0.276484	ns
Atom Swap-6 vs Atom Swap-6 SAS mask 2	MPO	1.226144	0.220722	ns
Atom Swap-6 vs Atom Swap-6 SAS mask 2	QED	1.265033	0.206443	ns
Atom Swap-6 vs Atom Swap-6 SAS mask 2	SAscore	-0.07769	0.938108	ns
Atom Swap-6 vs Atom Swap-6 FG mask	MPO	-4.6511	4.19E-06	****
Atom Swap-6 vs Atom Swap-6 FG mask	QED	3.197095	0.001475	**
Atom Swap-6 vs Atom Swap-6 FG mask	SAscore	0.326405	0.74425	ns
Atom Swap-6 SAS mask 0 vs Atom Swap-6 SAS mask 1	MPO	2.977675	0.003037	**
Atom Swap-6 SAS mask 0 vs Atom Swap-6 SAS mask 1	QED	-3.05334	0.002374	**
Atom Swap-6 SAS mask 0 vs Atom Swap-6 SAS mask 1	SAscore	1.266415	0.20592	ns
Atom Swap-6 SAS mask 0 vs Atom Swap-6 SAS mask 2	MPO	2.849979	0.004541	**
Atom Swap-6 SAS mask 0 vs Atom Swap-6 SAS mask 2	QED	0.432017	0.665906	ns
Atom Swap-6 SAS mask 0 vs Atom Swap-6 SAS mask 2	SAscore	0.119395	0.905008	ns
Atom Swap-6 SAS mask 0 vs Atom Swap-6 FG mask	MPO	-3.2507	0.001223	**
Atom Swap-6 SAS mask 0 vs Atom Swap-6 FG mask	QED	2.456591	0.014334	*
Atom Swap-6 SAS mask 0 vs Atom Swap-6 FG mask	SAscore	0.517517	0.605005	ns
Atom Swap-6 SAS mask 1 vs Atom Swap-6 SAS mask 2	MPO	-0.02685	0.978586	ns
Atom Swap-6 SAS mask 1 vs Atom Swap-6 SAS mask 2	QED	3.335641	0.000913	***
Atom Swap-6 SAS mask 1 vs Atom Swap-6 SAS mask 2	SAscore	-1.10063	0.271564	ns
Atom Swap-6 SAS mask 1 vs Atom Swap-6 FG mask	MPO	-6.21325	1.08E-09	****
Atom Swap-6 SAS mask 1 vs Atom Swap-6 FG mask	QED	5.536783	4.82E-08	****
Atom Swap-6 SAS mask 1 vs Atom Swap-6 FG mask	SAscore	-0.72582	0.468267	ns
Atom Swap-6 SAS mask 2 vs Atom Swap-6 FG mask	MPO	-6.00464	3.61E-09	****
Atom Swap-6 SAS mask 2 vs Atom Swap-6 FG mask	QED	1.903385	0.057538	ns
Atom Swap-6 SAS mask 2 vs Atom Swap-6 FG mask	SAscore	0.380822	0.703488	ns

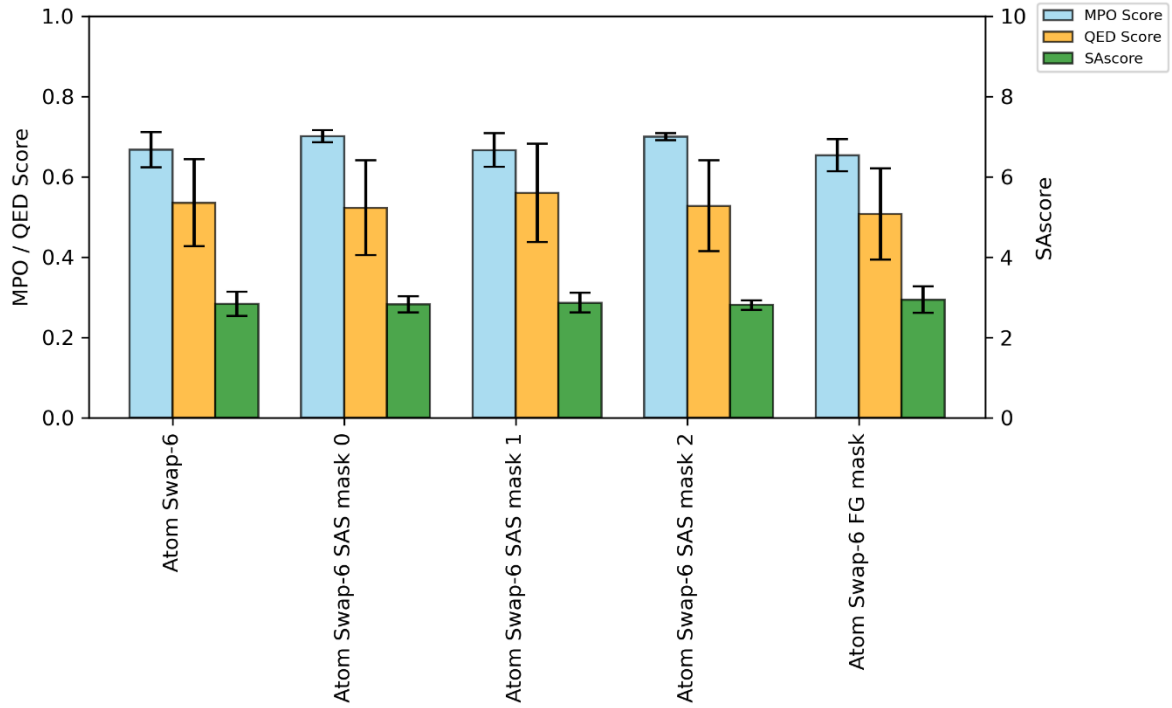


Figure 0-4. The Average Zaleplon MPO, SAscore, and QED Scores of the Molecules Generated by the RNN-LSTM Atom-Swap Augmented Model with and without Masking. The atom-swap with SAscore mask also included thresholds of 0, 1, and 2. The results shown are averaged across three repeats with standard deviation. The left y-axis represents the scale of values for the MPO and QED scores. The right y-axis represents the scale of values for the SAscores.

Table 0-8. Welch's t-test Results for the Pairwise Comparison of the augmented RNN-LSTM using Atom-Swap with and without Masking when tasked on the Zaleplon MPO. The significance indicates: **** (p <= 0.0001), *** (p <= 0.001), ** (p <= 0.01), * (p <= 0.05), and ns (p > 0.05).

Comparison	Score	t-statistic	p-value	Significance
Atom Swap-6 vs Atom Swap-6 SAS mask 0	MPO	-11.6919	1.53E-26	****
Atom Swap-6 vs Atom Swap-6 SAS mask 0	QED	1.243697	0.214216	ns
Atom Swap-6 vs Atom Swap-6 SAS mask 0	SAscore	0.58298	0.560197	ns
Atom Swap-6 vs Atom Swap-6 SAS mask 1	MPO	0.276731	0.782094	ns
Atom Swap-6 vs Atom Swap-6 SAS mask 1	QED	-2.47005	0.013808	*
Atom Swap-6 vs Atom Swap-6 SAS mask 1	SAscore	-1.27804	0.201843	ns
Atom Swap-6 vs Atom Swap-6 SAS mask 2	MPO	-11.6226	1.02E-25	****
Atom Swap-6 vs Atom Swap-6 SAS mask 2	QED	0.740636	0.459268	ns
Atom Swap-6 vs Atom Swap-6 SAS mask 2	SAscore	1.474012	0.141398	ns
Atom Swap-6 vs Atom Swap-6 FG mask	MPO	3.90748	0.000105	***
Atom Swap-6 vs Atom Swap-6 FG mask	QED	2.980413	0.003007	**
Atom Swap-6 vs Atom Swap-6 FG mask	SAscore	-3.6839	0.000253	***
Atom Swap-6 SAS mask 0 vs Atom Swap-6 SAS mask 1	MPO	13.45695	3.78E-34	****
Atom Swap-6 SAS mask 0 vs Atom Swap-6 SAS mask 1	QED	-3.52615	0.000459	***
Atom Swap-6 SAS mask 0 vs Atom Swap-6 SAS mask 1	SAscore	-2.27432	0.023343	*
Atom Swap-6 SAS mask 0 vs Atom Swap-6 SAS mask 2	MPO	1.351909	0.177218	ns
Atom Swap-6 SAS mask 0 vs Atom Swap-6 SAS mask 2	QED	-0.50034	0.617064	ns
Atom Swap-6 SAS mask 0 vs Atom Swap-6 SAS mask 2	SAscore	1.095586	0.273942	ns
Atom Swap-6 SAS mask 0 vs Atom Swap-6 FG mask	MPO	18.85466	1.81E-56	****
Atom Swap-6 SAS mask 0 vs Atom Swap-6 FG mask	QED	1.5357	0.125244	ns
Atom Swap-6 SAS mask 0 vs Atom Swap-6 FG mask	SAscore	-4.82287	1.89E-06	****
Atom Swap-6 SAS mask 1 vs Atom Swap-6 SAS mask 2	MPO	-13.5329	1.86E-33	****
Atom Swap-6 SAS mask 1 vs Atom Swap-6 SAS mask 2	QED	3.08788	0.002122	**
Atom Swap-6 SAS mask 1 vs Atom Swap-6 SAS mask 2	SAscore	3.788487	0.000172	***
Atom Swap-6 SAS mask 1 vs Atom Swap-6 FG mask	MPO	3.886798	0.000113	***
Atom Swap-6 SAS mask 1 vs Atom Swap-6 FG mask	QED	5.368857	1.14E-07	****
Atom Swap-6 SAS mask 1 vs Atom Swap-6 FG mask	SAscore	-2.93501	0.003482	**
Atom Swap-6 SAS mask 2 vs Atom Swap-6 FG mask	MPO	19.23011	2.80E-55	****
Atom Swap-6 SAS mask 2 vs Atom Swap-6 FG mask	QED	2.106754	0.035625	*
Atom Swap-6 SAS mask 2 vs Atom Swap-6 FG mask	SAscore	-6.19055	1.57E-09	****

Atom-Fusion

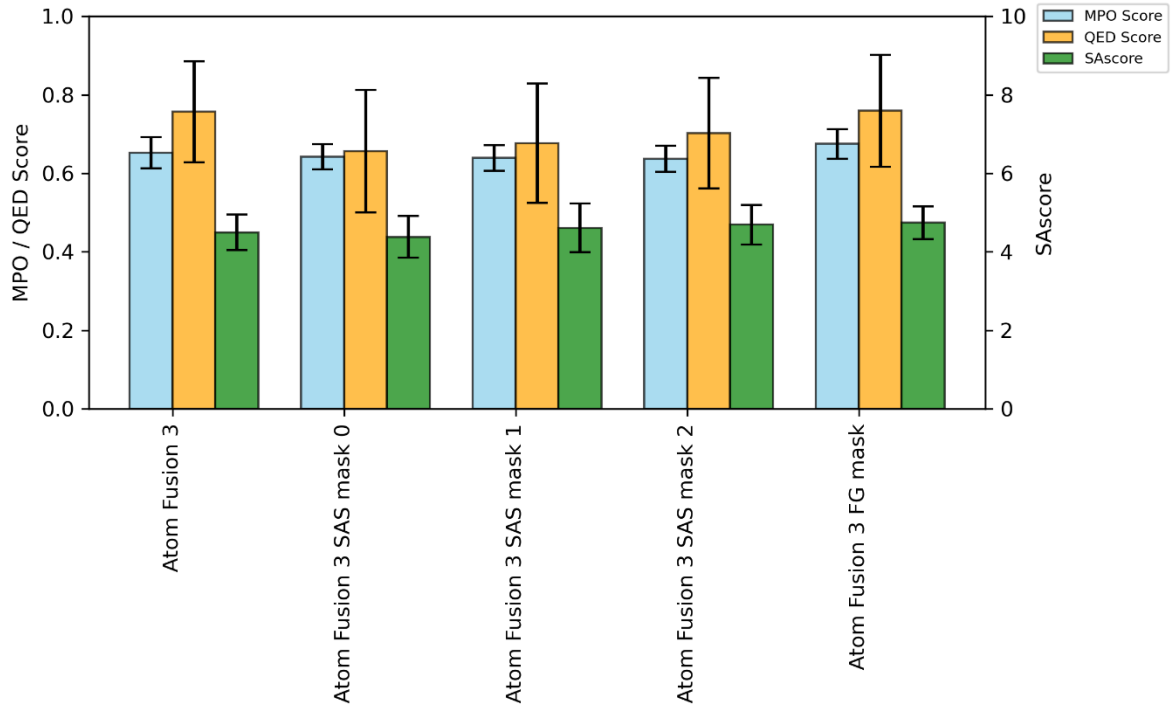


Figure 0-5. The Average Sitagliptin MPO, SAscore, and QED Scores of the Molecules Generated by the RNN-LSTM Atom-Fusion Augmented Model with and without Masking. The atom-fusion with SAscore mask also included thresholds of 0, 1, and 2. The results shown are averaged across three repeats with standard deviation. The left y-axis represents the scale of values for the MPO and QED scores. The right y-axis represents the scale of values for the SAscores.

Table 0-9. Welch's t-test Results for the Pairwise Comparison of the augmented RNN-LSTM using Atom-Fusion with and without Masking when tasked on the Sitagliptin MPO. The significance indicates: **** (p <= 0.0001), *** (p <= 0.001), ** (p <= 0.01), * (p <= 0.05), and ns (p > 0.05).

Comparison	Score	t-statistic	p-value	Significance
Atom Fusion 3 vs Atom Fusion 3 SAS mask 0	MPO	3.277504	0.001112	**
Atom Fusion 3 vs Atom Fusion 3 SAS mask 0	QED	8.5321	1.32E-16	****
Atom Fusion 3 vs Atom Fusion 3 SAS mask 0	SAscore	2.828176	0.004844	**
Atom Fusion 3 vs Atom Fusion 3 SAS mask 1	MPO	4.410807	1.23E-05	****
Atom Fusion 3 vs Atom Fusion 3 SAS mask 1	QED	6.894093	1.46E-11	****
Atom Fusion 3 vs Atom Fusion 3 SAS mask 1	SAscore	-2.49647	0.012846	*
Atom Fusion 3 vs Atom Fusion 3 SAS mask 2	MPO	5.184307	3.01E-07	****
Atom Fusion 3 vs Atom Fusion 3 SAS mask 2	QED	4.96514	8.99E-07	****
Atom Fusion 3 vs Atom Fusion 3 SAS mask 2	SAscore	-4.92962	1.07E-06	****
Atom Fusion 3 vs Atom Fusion 3 FG mask	MPO	-7.00957	6.52E-12	****
Atom Fusion 3 vs Atom Fusion 3 FG mask	QED	-0.2308	0.817549	ns
Atom Fusion 3 vs Atom Fusion 3 FG mask	SAscore	-6.86047	1.74E-11	****
Atom Fusion 3 SAS mask 0 vs Atom Fusion 3 SAS mask 1	MPO	1.299207	0.194389	ns
Atom Fusion 3 SAS mask 0 vs Atom Fusion 3 SAS mask 1	QED	-1.57605	0.115558	ns
Atom Fusion 3 SAS mask 0 vs Atom Fusion 3 SAS mask 1	SAscore	-4.76893	2.36E-06	****
Atom Fusion 3 SAS mask 0 vs Atom Fusion 3 SAS mask 2	MPO	2.167382	0.030604	*
Atom Fusion 3 SAS mask 0 vs Atom Fusion 3 SAS mask 2	QED	-3.73479	0.000206	***
Atom Fusion 3 SAS mask 0 vs Atom Fusion 3 SAS mask 2	SAscore	-7.28805	1.02E-12	****
Atom Fusion 3 SAS mask 0 vs Atom Fusion 3 FG mask	MPO	-11.1403	3.06E-26	****
Atom Fusion 3 SAS mask 0 vs Atom Fusion 3 FG mask	QED	-8.3784	4.00E-16	****
Atom Fusion 3 SAS mask 0 vs Atom Fusion 3 FG mask	SAscore	-9.2139	6.50E-19	****
Atom Fusion 3 SAS mask 1 vs Atom Fusion 3 SAS mask 2	MPO	0.859904	0.390193	ns
Atom Fusion 3 SAS mask 1 vs Atom Fusion 3 SAS mask 2	QED	-2.10763	0.035492	*
Atom Fusion 3 SAS mask 1 vs Atom Fusion 3 SAS mask 2	SAscore	-1.75041	0.0806	ns
Atom Fusion 3 SAS mask 1 vs Atom Fusion 3 FG mask	MPO	-12.2542	7.69E-31	****
Atom Fusion 3 SAS mask 1 vs Atom Fusion 3 FG mask	QED	-6.80623	2.50E-11	****
Atom Fusion 3 SAS mask 1 vs Atom Fusion 3 FG mask	SAscore	-3.05064	0.002405	**
Atom Fusion 3 SAS mask 2 vs Atom Fusion 3 FG mask	MPO	-13.0581	2.28E-34	****
Atom Fusion 3 SAS mask 2 vs Atom Fusion 3 FG mask	QED	-4.94878	9.73E-07	****
Atom Fusion 3 SAS mask 2 vs Atom Fusion 3 FG mask	SAscore	-1.36675	0.172235	ns

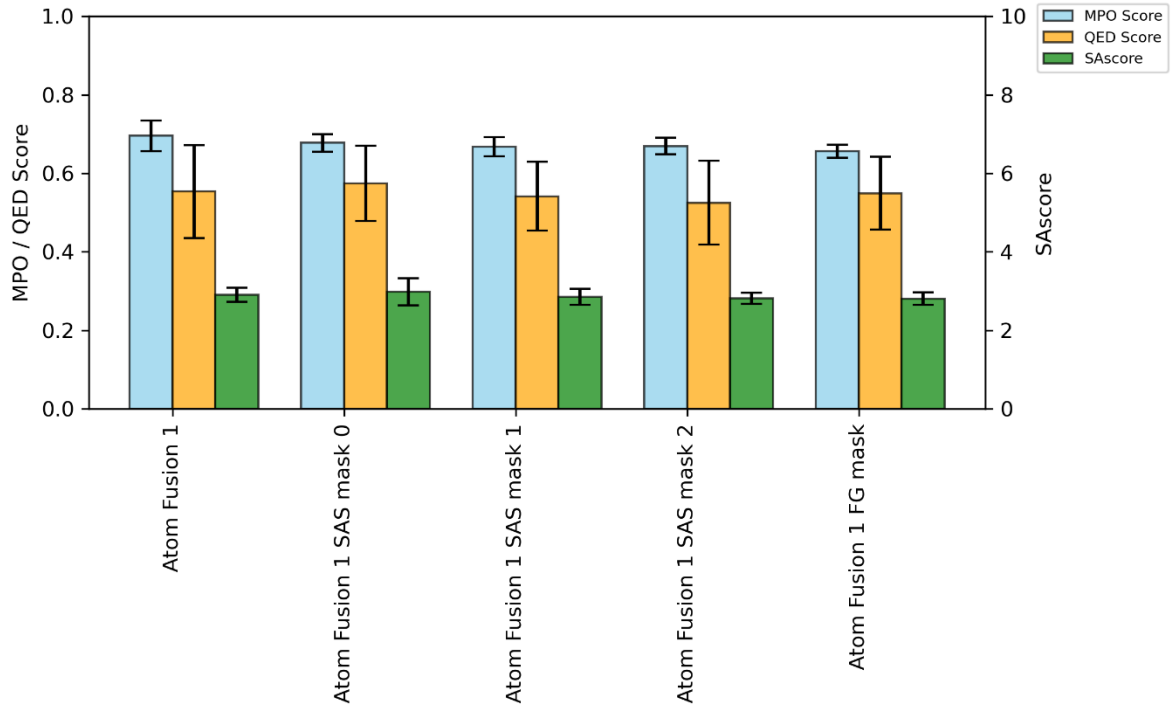


Figure 0-6. he Average Zaleplon MPO, SAscore, and QED Scores of the Molecules Generated by the RNN-LSTM Atom-Fusion Augmented Model with and without Masking. The atom-fusion with SAscore mask also included thresholds of 0, 1, and 2. The results shown are averaged across three repeats with standard deviation. The left y-axis represents the scale of values for the MPO and QED scores. The right y-axis represents the scale of values for the SAscores.

Table 0-10. Welch's t-test Results for the Pairwise Comparison of the augmented RNN-LSTM using Atom-Fusion with and without Masking when tasked on the Zaleplon MPO. The significance indicates: **** (p <= 0.0001), *** (p <= 0.001), ** (p <= 0.01), * (p <= 0.05), and ns (p > 0.05).

Comparison	Score	t-statistic	p-value	Significance
Atom Fusion 1 vs Atom Fusion 1 SAS mask 0	MPO	6.744539	4.47E-11	****
Atom Fusion 1 vs Atom Fusion 1 SAS mask 0	QED	-2.39604	0.016892	*
Atom Fusion 1 vs Atom Fusion 1 SAS mask 0	SAscore	-3.2364	0.0013	**
Atom Fusion 1 vs Atom Fusion 1 SAS mask 1	MPO	9.956309	1.88E-21	****
Atom Fusion 1 vs Atom Fusion 1 SAS mask 1	QED	1.34716	0.178495	ns
Atom Fusion 1 vs Atom Fusion 1 SAS mask 1	SAscore	3.264646	0.001171	**
Atom Fusion 1 vs Atom Fusion 1 SAS mask 2	MPO	10.10112	8.05E-22	****
Atom Fusion 1 vs Atom Fusion 1 SAS mask 2	QED	3.063688	0.002289	**
Atom Fusion 1 vs Atom Fusion 1 SAS mask 2	SAscore	6.482653	1.98E-10	****
Atom Fusion 1 vs Atom Fusion 1 FG mask	MPO	15.9906	4.67E-45	****
Atom Fusion 1 vs Atom Fusion 1 FG mask	QED	0.463521	0.643171	ns
Atom Fusion 1 vs Atom Fusion 1 FG mask	SAscore	6.873441	1.64E-11	****
Atom Fusion 1 SAS mask 0 vs Atom Fusion 1 SAS mask 1	MPO	4.816154	1.94E-06	****
Atom Fusion 1 SAS mask 0 vs Atom Fusion 1 SAS mask 1	QED	4.180893	3.39E-05	****
Atom Fusion 1 SAS mask 0 vs Atom Fusion 1 SAS mask 1	SAscore	5.349648	1.35E-07	****
Atom Fusion 1 SAS mask 0 vs Atom Fusion 1 SAS mask 2	MPO	4.756326	2.50E-06	****
Atom Fusion 1 SAS mask 0 vs Atom Fusion 1 SAS mask 2	QED	5.865896	7.67E-09	****
Atom Fusion 1 SAS mask 0 vs Atom Fusion 1 SAS mask 2	SAscore	7.385739	8.78E-13	****
Atom Fusion 1 SAS mask 0 vs Atom Fusion 1 FG mask	MPO	13.45671	7.17E-36	****
Atom Fusion 1 SAS mask 0 vs Atom Fusion 1 FG mask	QED	3.188282	0.00151	**
Atom Fusion 1 SAS mask 0 vs Atom Fusion 1 FG mask	SAscore	7.703054	9.18E-14	****
Atom Fusion 1 SAS mask 1 vs Atom Fusion 1 SAS mask 2	MPO	-0.55768	0.57732	ns
Atom Fusion 1 SAS mask 1 vs Atom Fusion 1 SAS mask 2	QED	1.973814	0.048934	*
Atom Fusion 1 SAS mask 1 vs Atom Fusion 1 SAS mask 2	SAscore	2.178595	0.029898	*
Atom Fusion 1 SAS mask 1 vs Atom Fusion 1 FG mask	MPO	6.670228	7.97E-11	****
Atom Fusion 1 SAS mask 1 vs Atom Fusion 1 FG mask	QED	-0.98338	0.325879	ns
Atom Fusion 1 SAS mask 1 vs Atom Fusion 1 FG mask	SAscore	2.75503	0.006096	**
Atom Fusion 1 SAS mask 2 vs Atom Fusion 1 FG mask	MPO	8.322635	7.55E-16	****
Atom Fusion 1 SAS mask 2 vs Atom Fusion 1 FG mask	QED	-2.8861	0.004056	**
Atom Fusion 1 SAS mask 2 vs Atom Fusion 1 FG mask	SAscore	0.83612	0.403457	ns

Chemistry-based Data Augmentation

Graph-based Operation

Table 0-11. The Sitagliptin MPO, SAscore, and QED Scores of the Molecules Generated by the Baselines (RNN-LSTM and Graph-GA) and the RNN-LSTM Augmented with Graph-Fusion at the subset loop = 5 point. The results are averaged across the 300 molecules generated across triplicate repeats. Results in bold indicate an improved and significant score compared to the RNN-LSTM baseline. The p-value was obtained using a pairwise comparison t-test against the RNN-LSTM (pv RNN) and Graph-GA (pv GA). The full result for the statistical testing is in Appendix Table 0-12. The p-value indicates levels of significance: **** (p <= 0.0001), *** (p <= 0.001), ** (p <= 0.01), * (p <= 0.05), and ns (p > 0.05).

Model	MPO	pv RNN	pv GA	QED	pv RNN	pv GA	SAscore	pv RNN	pv GA
RNN-LSTM	0.452±0.033	-	****	0.709±0.124	-	****	2.780±0.577	-	****
Graph-GA	**0.763±0.046**	****	-	0.506±0.142	****	-	5.626±0.380	****	-
Graph Fus-1	**0.867±0.021**	****	****	0.356±0.103	****	****	6.269±0.260	****	****
Graph Fus-2	**0.823±0.028**	****	****	0.204±0.156	****	****	6.030±0.501	****	****
Graph Fus-3	**0.799±0.024**	****	****	0.251±0.184	****	****	6.148±0.507	****	****

Table 0-12. Welch's t-test Results for the Pairwise Comparison of the baselines (RNN-LSTM and Graph-GA) and the augmented RNN-LSTM (with Graph-Fusion) when tasked on the Sitagliptin MPO. The significance indicates: **** (p <= 0.0001), *** (p <= 0.001), ** (p <= 0.01), * (p <= 0.05), and ns (p > 0.05).

Comparison	Score	t-statistic	p-value	Significance
RNN-LSTM vs Graph-GA	MPO	-87.1705	3.14E-299	****
RNN-LSTM vs Graph-GA	QED	16.67923	5.88E-49	****
RNN-LSTM vs Graph-GA	SAscore	-60.539	6.20E-170	****
RNN-LSTM vs Graph Fus-1	MPO	-156.607	9.52E-287	****
RNN-LSTM vs Graph Fus-1	QED	32.94918	6.28E-110	****
RNN-LSTM vs Graph Fus-1	SAscore	-78.9635	4.86E-175	****
RNN-LSTM vs Graph Fus-2	MPO	-130.232	1.59E-306	****
RNN-LSTM vs Graph Fus-2	QED	39.76986	1.47E-152	****
RNN-LSTM vs Graph Fus-2	SAscore	-64.1807	8.84E-202	****
RNN-LSTM vs Graph Fus-3	MPO	-127.576	6.47E-276	****
RNN-LSTM vs Graph Fus-3	QED	33.05827	7.57E-127	****
RNN-LSTM vs Graph Fus-3	SAscore	-66.2498	1.12E-207	****
Graph-GA vs Graph Fus-1	MPO	-35.5563	1.46E-128	****
Graph-GA vs Graph Fus-1	QED	14.82804	5.17E-42	****
Graph-GA vs Graph Fus-1	SAscore	-24.1814	1.62E-87	****
Graph-GA vs Graph Fus-2	MPO	-19.3767	1.62E-62	****
Graph-GA vs Graph Fus-2	QED	24.75514	1.91E-93	****
Graph-GA vs Graph Fus-2	SAscore	-11.1175	4.50E-26	****
Graph-GA vs Graph Fus-3	MPO	-12.0342	4.59E-29	****
Graph-GA vs Graph Fus-3	QED	19.05481	7.65E-63	****
Graph-GA vs Graph Fus-3	SAscore	-14.259	1.58E-39	****
Graph Fus-1 vs Graph Fus-2	MPO	21.83875	6.89E-77	****
Graph Fus-1 vs Graph Fus-2	QED	14.09174	2.20E-38	****
Graph Fus-1 vs Graph Fus-2	SAscore	7.349103	9.51E-13	****
Graph Fus-1 vs Graph Fus-3	MPO	37.42213	4.58E-158	****
Graph Fus-1 vs Graph Fus-3	QED	8.687933	6.20E-17	****
Graph Fus-1 vs Graph Fus-3	SAscore	3.689317	0.000253	***
Graph Fus-2 vs Graph Fus-3	MPO	11.56363	5.53E-28	****
Graph Fus-2 vs Graph Fus-3	QED	-3.34751	0.000868	***
Graph Fus-2 vs Graph Fus-3	SAscore	-2.86957	0.004256	**

Table 0-13. The Zaleplon MPO, SAscore, and QED Scores of the Molecules Generated by the Baselines (RNN-LSTM and Graph-GA) and the RNN-LSTM Augmented with Graph-Fusion at the subset loop = 5 point. The results are averaged across the 300 molecules generated across triplicate repeats. Results in bold indicate an improved and significant score compared to the RNN-LSTM baseline. The p-value was obtained using a pairwise comparison t-test against the RNN-LSTM (pv RNN) and Graph-GA (pv GA). The full result for the statistical testing is in Appendix Table 0-14. The p-value indicates levels of significance: **** (p <= 0.0001), *** (p <= 0.001), ** (p <= 0.01), * (p <= 0.05), and ns (p > 0.05).

Model	MPO	pv RNN	pv GA	QED	pv RNN	pv GA	SAscore	pv RNN	pv GA
RNN-LSTM	0.562±0.019	-	****	0.688±0.092	-	****	2.323±0.179	-	****
Graph-GA	**0.681±0.021**	****	-	0.561±0.110	****	-	2.774±0.248	****	-
Graph Fus-1	**0.714±0.016**	****	****	0.486±0.128	****	****	3.106±0.295	****	****
Graph Fus-2	**0.715±0.016**	****	****	0.502±0.112	****	****	3.069±0.296	****	****
Graph Fus-3	**0.707±0.011**	****	****	0.483±0.111	****	****	3.021±0.267	****	****

Table 0-14. Welch's t-test Results for the Pairwise Comparison of the baselines (RNN-LSTM and Graph-GA) and the augmented RNN-LSTM (with Graph-Fusion) when tasked on the Zaleplon MPO. The significance indicates: **** (p <= 0.0001), *** (p <= 0.001), ** (p <= 0.01), * (p <= 0.05), and ns (p > 0.05).

Comparison	Score	t-statistic	p-value	Significance
RNN-LSTM vs Graph-GA	MPO	-65.3377	1.16E-229	****
RNN-LSTM vs Graph-GA	QED	13.62829	2.27E-35	****
RNN-LSTM vs Graph-GA	SAscore	-22.4982	2.63E-72	****
RNN-LSTM vs Graph Fus-1	MPO	-100.666	0	****
RNN-LSTM vs Graph Fus-1	QED	21.17848	7.51E-72	****
RNN-LSTM vs Graph Fus-1	SAscore	-37.7066	1.01E-143	****
RNN-LSTM vs Graph Fus-2	MPO	-103.758	0	****
RNN-LSTM vs Graph Fus-2	QED	21.435	2.55E-74	****
RNN-LSTM vs Graph Fus-2	SAscore	-36.1823	4.45E-139	****
RNN-LSTM vs Graph Fus-3	MPO	-109.008	4.6964317187057e-310	****
RNN-LSTM vs Graph Fus-3	QED	23.17035	5.25E-82	****
RNN-LSTM vs Graph Fus-3	SAscore	-35.4899	3.51E-135	****
Graph-GA vs Graph Fus-1	MPO	-19.5663	1.58E-60	****
Graph-GA vs Graph Fus-1	QED	7.002085	8.28E-12	****
Graph-GA vs Graph Fus-1	SAscore	-13.6542	2.98E-36	****
Graph-GA vs Graph Fus-2	MPO	-20.6662	1.02E-64	****
Graph-GA vs Graph Fus-2	QED	5.958257	4.98E-09	****
Graph-GA vs Graph Fus-2	SAscore	-12.2089	3.46E-30	****
Graph-GA vs Graph Fus-3	MPO	-17.1854	3.62E-47	****
Graph-GA vs Graph Fus-3	QED	7.73133	6.54E-14	****
Graph-GA vs Graph Fus-3	SAscore	-10.5481	1.58E-23	****
Graph Fus-1 vs Graph Fus-2	MPO	-0.95539	0.339789	ns
Graph Fus-1 vs Graph Fus-2	QED	-1.54262	0.123495	ns
Graph Fus-1 vs Graph Fus-2	SAscore	1.474195	0.140984	ns
Graph Fus-1 vs Graph Fus-3	MPO	5.685955	2.23E-08	****
Graph Fus-1 vs Graph Fus-3	QED	0.297073	0.766524	ns
Graph Fus-1 vs Graph Fus-3	SAscore	3.563402	0.000398	***
Graph Fus-2 vs Graph Fus-3	MPO	7.020829	6.99E-12	****
Graph Fus-2 vs Graph Fus-3	QED	1.970305	0.049301	*
Graph Fus-2 vs Graph Fus-3	SAscore	2.049558	0.040877	*

Medicinal-Chemistry Transformation

Bioisosteric Substitution

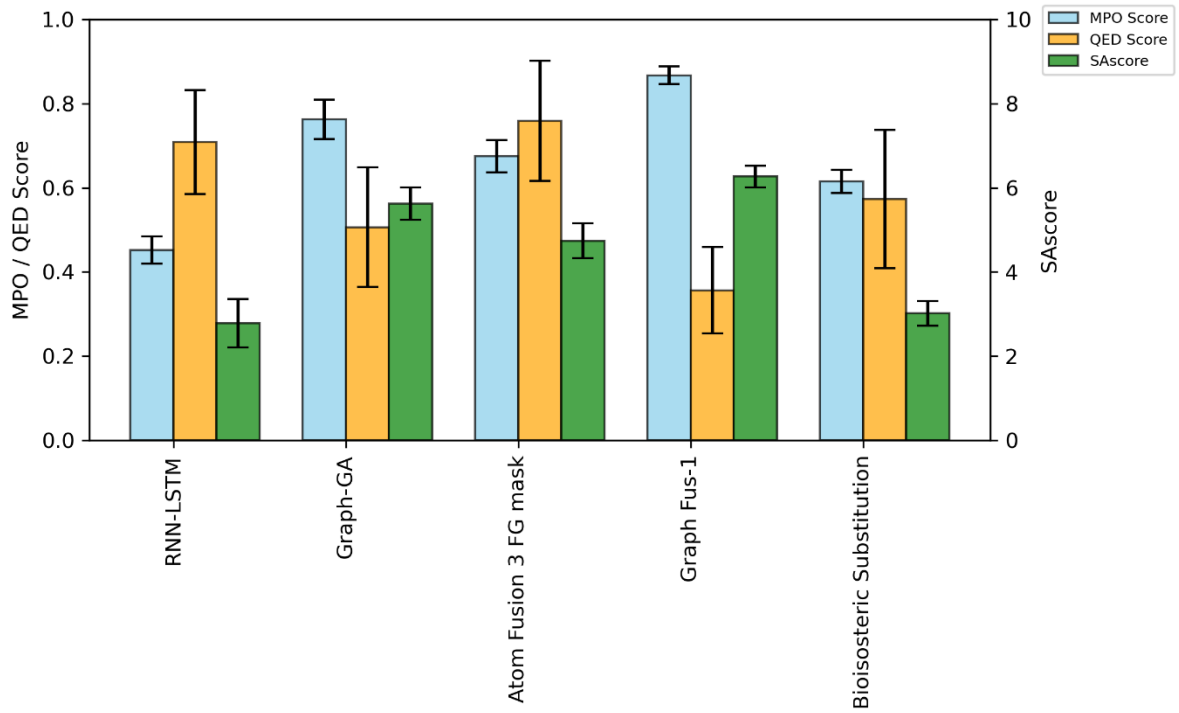


Figure 0-7. The Average Sitagliptin MPO Task, SAscore, and QED Scores of the Molecules Generated by the RNN-LSTM Bioisosteric Substitution Augmented Model Compared to Previous Operators and Baselines. The results shown are averaged across three repeats. The left y-axis represents the scale of values for the MPO and QED scores. The right y-axis represents the scale of values for the SAscores.

Table 0-15. Welch's t-test Results for the Pairwise Comparison of the baselines (RNN-LSTM and Graph-GA) and the augmented RNN-LSTM (Atom-fusion 3 with FG mask, Graph—fusion 1, and bioisosteric substitution) when tasked on the Sitagliptin MPO. The significance indicates: **** (p <= 0.0001), *** (p <= 0.001), ** (p <= 0.01), * (p <= 0.05), and ns (p > 0.05).

Comparison	Score	t-statistic	p-value	Significance
RNN-LSTM vs Graph-GA	MPO	-87.1705	3.14E-299	****
RNN-LSTM vs Graph-GA	QED	16.67923	5.88E-49	****
RNN-LSTM vs Graph-GA	SAscore	-60.539	6.20E-170	****
RNN-LSTM vs Atom Fusion 3 FG mask	MPO	-68.9962	1.42E-242	****
RNN-LSTM vs Atom Fusion 3 FG mask	QED	-4.17251	3.62E-05	****
RNN-LSTM vs Atom Fusion 3 FG mask	SAscore	-40.8864	1.17E-128	****
RNN-LSTM vs Graph Fus-1	MPO	-156.607	9.52E-287	****
RNN-LSTM vs Graph Fus-1	QED	32.94918	6.28E-110	****
RNN-LSTM vs Graph Fus-1	SAscore	-78.9635	4.86E-175	****
RNN-LSTM vs Bioisosteric Substitution	MPO	-54.5017	1.36E-180	****
RNN-LSTM vs Bioisosteric Substitution	QED	9.490732	2.12E-19	****
RNN-LSTM vs Bioisosteric Substitution	SAscore	-5.17613	4.35E-07	****
Graph-GA vs Atom Fusion 3 FG mask	MPO	25.28362	1.70E-95	****
Graph-GA vs Atom Fusion 3 FG mask	QED	-21.7116	1.75E-77	****
Graph-GA vs Atom Fusion 3 FG mask	SAscore	27.23235	1.65E-106	****
Graph-GA vs Graph Fus-1	MPO	-35.5563	1.46E-128	****
Graph-GA vs Graph Fus-1	QED	14.82804	5.17E-42	****
Graph-GA vs Graph Fus-1	SAscore	-24.1814	1.62E-87	****
Graph-GA vs Bioisosteric Substitution	MPO	45.28207	1.69E-178	****
Graph-GA vs Bioisosteric Substitution	QED	-4.83972	1.82E-06	****
Graph-GA vs Bioisosteric Substitution	SAscore	87.93758	6.10179585545e-312	****
Atom Fusion 3 FG mask vs Graph Fus-1	MPO	-76.1134	1.47E-264	****
Atom Fusion 3 FG mask vs Graph Fus-1	QED	39.70292	2.17E-162	****
Atom Fusion 3 FG mask vs Graph Fus-1	SAscore	-54.0855	2.94E-211	****
Atom Fusion 3 FG mask vs Bioisosteric Substitution	MPO	20.65436	1.75E-69	****
Atom Fusion 3 FG mask vs Bioisosteric Substitution	QED	13.42527	1.56E-34	****
Atom Fusion 3 FG mask vs Bioisosteric Substitution	SAscore	55.31113	9.96E-219	****
Graph Fus-1 vs Bioisosteric Substitution	MPO	113.509	1.27E-302	****
Graph Fus-1 vs Bioisosteric Substitution	QED	-17.2337	3.44E-48	****
Graph Fus-1 vs Bioisosteric Substitution	SAscore	130.3812	0	****

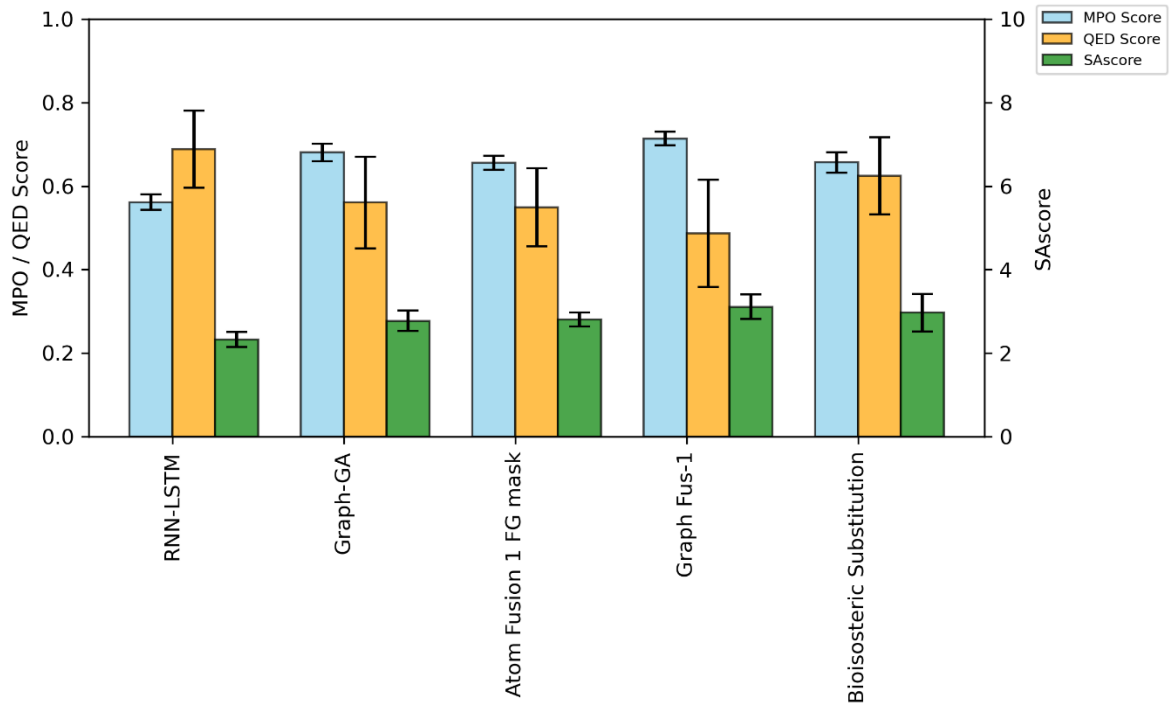


Figure 0-8. The Average Zaleplon MPO Task, SAscore, and QED Scores of the Molecules Generated by the RNN-LSTM Bioisosteric Substitution Augmented Model Compared to Previous Operators and Baselines. The results shown are averaged across three repeats. The left y-axis represents the scale of values for the MPO and QED scores. The right y-axis represents the scale of values for the SAscores.

Table 0-16. Welch's t-test Results for the Pairwise Comparison of the baselines (RNN-LSTM and Graph-GA) and the augmented RNN-LSTM (Atom-fusion 3 with FG mask, Graph—fusion 1, and bioisosteric substitution) when tasked on the Zaleplon MPO. The significance indicates: **** (p <= 0.0001), *** (p <= 0.001), ** (p <= 0.01), * (p <= 0.05), and ns (p > 0.05).

Comparison	Score	t-statistic	p-value	Significance
RNN-LSTM vs Graph-GA	MPO	-65.3377	1.16E-229	****
RNN-LSTM vs Graph-GA	QED	13.62829	2.27E-35	****
RNN-LSTM vs Graph-GA	SAscore	-22.4982	2.63E-72	****
RNN-LSTM vs Atom Fusion 1 FG mask	MPO	-61.4503	1.11E-240	****
RNN-LSTM vs Atom Fusion 1 FG mask	QED	17.40177	4.62E-54	****
RNN-LSTM vs Atom Fusion 1 FG mask	SAscore	-32.6682	1.54E-128	****
RNN-LSTM vs Graph Fus-1	MPO	-100.666	0	****
RNN-LSTM vs Graph Fus-1	QED	21.17848	7.51E-72	****
RNN-LSTM vs Graph Fus-1	SAscore	-37.7066	1.01E-143	****
RNN-LSTM vs Bioisosteric Substitution	MPO	-47.6074	3.42E-169	****
RNN-LSTM vs Bioisosteric Substitution	QED	7.561302	2.13E-13	****
RNN-LSTM vs Bioisosteric Substitution	SAscore	-20.1692	1.95E-56	****
Graph-GA vs Atom Fusion 1 FG mask	MPO	14.46323	1.29E-38	****
Graph-GA vs Atom Fusion 1 FG mask	QED	1.262058	0.207616	ns
Graph-GA vs Atom Fusion 1 FG mask	SAscore	-1.6116	0.107927	ns
Graph-GA vs Graph Fus-1	MPO	-19.5663	1.58E-60	****
Graph-GA vs Graph Fus-1	QED	7.002085	8.28E-12	****
Graph-GA vs Graph Fus-1	SAscore	-13.6542	2.98E-36	****
Graph-GA vs Bioisosteric Substitution	MPO	11.13368	1.81E-25	****
Graph-GA vs Bioisosteric Substitution	QED	-6.58408	1.35E-10	****
Graph-GA vs Bioisosteric Substitution	SAscore	-5.63833	3.58E-08	****
Atom Fusion 1 FG mask vs Graph Fus-1	MPO	-41.7869	2.69E-173	****
Atom Fusion 1 FG mask vs Graph Fus-1	QED	6.625389	8.80E-11	****
Atom Fusion 1 FG mask vs Graph Fus-1	SAscore	-14.9111	5.83E-41	****
Atom Fusion 1 FG mask vs Bioisosteric Substitution	MPO	-0.53647	0.591952	ns
Atom Fusion 1 FG mask vs Bioisosteric Substitution	QED	-9.00881	5.15E-18	****
Atom Fusion 1 FG mask vs Bioisosteric Substitution	SAscore	-5.14614	5.16E-07	****
Graph Fus-1 vs Bioisosteric Substitution	MPO	30.26965	4.04E-102	****
Graph Fus-1 vs Bioisosteric Substitution	QED	-14.031	6.60E-38	****
Graph Fus-1 vs Bioisosteric Substitution	SAscore	3.959731	9.03E-05	****

Multimodal Fusion Operator

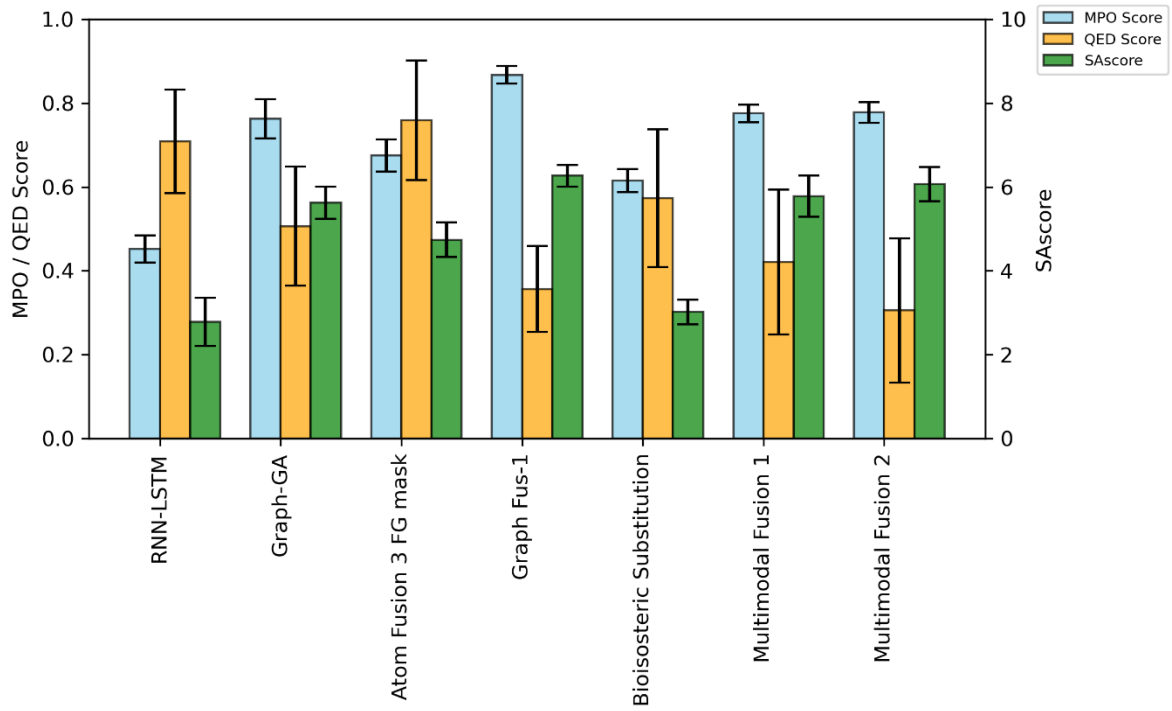


Figure 0-9. The Average Sitagliptin MPO Task, SAscore, and QED Scores of the Molecules Generated by the RNN-LSTM Augmented with the Multimodal-Fusion 1 and 2 Operators Compared to Previous Operators and Baselines. The results shown are averaged across three repeats. The left y-axis represents the scale of values for the MPO and QED scores. The right y-axis represents the scale of values for the SAscores.

Table 0-17. Welch's t-test Results for the Pairwise Comparison of the baselines (RNN-LSTM and Graph-GA) and the augmented RNN-LSTM (Atom-fusion 3 with FG mask, Graph—fusion 1, Bioisosteric Substitution, Multimodal-fusion 1, Multimodal-fusion 2) when tasked on the Sitagliptin MPO. The significance indicates: **** (p <= 0.0001), *** (p <= 0.001), ** (p <= 0.01), * (p <= 0.05), and ns (p > 0.05).

Comparison	Score	t-statistic	p-value	Significance
RNN-LSTM vs Graph-GA	MPO	-87.1705	3.14E-299	****
RNN-LSTM vs Graph-GA	QED	16.67923	5.88E-49	****
RNN-LSTM vs Graph-GA	SAscore	-60.539	6.20E-170	****
RNN-LSTM vs Atom Fusion 3 FG mask	MPO	-68.9962	1.42E-242	****
RNN-LSTM vs Atom Fusion 3 FG mask	QED	-4.17251	3.62E-05	****
RNN-LSTM vs Atom Fusion 3 FG mask	SAscore	-40.8864	1.17E-128	****
RNN-LSTM vs Graph Fus-1	MPO	-156.607	9.52E-287	****
RNN-LSTM vs Graph Fus-1	QED	32.94918	6.28E-110	****
RNN-LSTM vs Graph Fus-1	SAscore	-78.9635	4.86E-175	****
RNN-LSTM vs Bioisosteric Substitution	MPO	-54.5017	1.36E-180	****
RNN-LSTM vs Bioisosteric Substitution	QED	9.490732	2.12E-19	****
RNN-LSTM vs Bioisosteric Substitution	SAscore	-5.17613	4.35E-07	****
RNN-LSTM vs Multimodal Fusion 1	MPO	-122.549	2.82E-253	****
RNN-LSTM vs Multimodal Fusion 1	QED	21.52688	1.17E-72	****
RNN-LSTM vs Multimodal Fusion 1	SAscore	-59.4926	3.54E-189	****
RNN-LSTM vs Multimodal Fusion 2	MPO	-118.443	4.67E-273	****
RNN-LSTM vs Multimodal Fusion 2	QED	30.26535	9.24E-114	****
RNN-LSTM vs Multimodal Fusion 2	SAscore	-68.677	6.87E-192	****
Graph-GA vs Atom Fusion 3 FG mask	MPO	25.28362	1.70E-95	****
Graph-GA vs Atom Fusion 3 FG mask	QED	-21.7116	1.75E-77	****
Graph-GA vs Atom Fusion 3 FG mask	SAscore	27.23235	1.65E-106	****
Graph-GA vs Graph Fus-1	MPO	-35.5563	1.46E-128	****
Graph-GA vs Graph Fus-1	QED	14.82804	5.17E-42	****
Graph-GA vs Graph Fus-1	SAscore	-24.1814	1.62E-87	****
Graph-GA vs Bioisosteric Substitution	MPO	45.28207	1.69E-178	****
Graph-GA vs Bioisosteric Substitution	QED	-4.83972	1.82E-06	****
Graph-GA vs Bioisosteric Substitution	SAscore	87.93758	6.10179585545e-312	****
Graph-GA vs Multimodal Fusion 1	MPO	-4.43998	1.16E-05	****
Graph-GA vs Multimodal Fusion 1	QED	6.635736	7.45E-11	****
Graph-GA vs Multimodal Fusion 1	SAscore	-4.16823	3.55E-05	****
Graph-GA vs Multimodal Fusion 2	MPO	-5.08298	5.43E-07	****

Graph-GA vs Multimodal Fusion 2	QED	15.61963	3.83E-46	****
Graph-GA vs Multimodal Fusion 2	SAscore	-13.6174	5.98E-37	****
Atom Fusion 3 FG mask vs Graph Fus-1	MPO	-76.1134	1.47E-264	****
Atom Fusion 3 FG mask vs Graph Fus-1	QED	39.70292	2.17E-162	****
Atom Fusion 3 FG mask vs Graph Fus-1	SAscore	-54.0855	2.94E-211	****
Atom Fusion 3 FG mask vs Bioisosteric Substitution	MPO	20.65436	1.75E-69	****
Atom Fusion 3 FG mask vs Bioisosteric Substitution	QED	13.42527	1.56E-34	****
Atom Fusion 3 FG mask vs Bioisosteric Substitution	SAscore	55.31113	9.96E-219	****
Atom Fusion 3 FG mask vs Multimodal Fusion 1	MPO	-40.055	5.38E-152	****
Atom Fusion 3 FG mask vs Multimodal Fusion 1	QED	26.15456	4.94E-100	****
Atom Fusion 3 FG mask vs Multimodal Fusion 1	SAscore	-27.8365	5.64E-109	****
Atom Fusion 3 FG mask vs Multimodal Fusion 2	MPO	-39.1858	5.84E-156	****
Atom Fusion 3 FG mask vs Multimodal Fusion 2	QED	35.20823	5.88E-146	****
Atom Fusion 3 FG mask vs Multimodal Fusion 2	SAscore	-39.3243	7.74E-168	****
Graph Fus-1 vs Bioisosteric Substitution	MPO	113.509	1.27E-302	****
Graph Fus-1 vs Bioisosteric Substitution	QED	-17.2337	3.44E-48	****
Graph Fus-1 vs Bioisosteric Substitution	SAscore	130.3812	0	****
Graph Fus-1 vs Multimodal Fusion 1	MPO	53.45359	6.70E-230	****
Graph Fus-1 vs Multimodal Fusion 1	QED	-5.54884	4.73E-08	****
Graph Fus-1 vs Multimodal Fusion 1	SAscore	15.35125	3.94E-43	****
Graph Fus-1 vs Multimodal Fusion 2	MPO	47.4132	2.17E-202	****
Graph Fus-1 vs Multimodal Fusion 2	QED	4.41328	1.25E-05	****
Graph Fus-1 vs Multimodal Fusion 2	SAscore	7.239188	1.69E-12	****
Bioisosteric Substitution vs Multimodal Fusion 1	MPO	-72.6672	6.05E-228	****
Bioisosteric Substitution vs Multimodal Fusion 1	QED	10.22065	2.53E-22	****
Bioisosteric Substitution vs Multimodal Fusion 1	SAscore	-79.4357	2.54E-285	****

Bioisosteric Substitution vs Multimodal Fusion 2	MPO	-69.5754	2.64E-239	****
Bioisosteric Substitution vs Multimodal Fusion 2	QED	17.99345	1.11E-55	****
Bioisosteric Substitution vs Multimodal Fusion 2	SAscore	-98.4308	0	****
Multimodal Fusion 1 vs Multimodal Fusion 2	MPO	-1.28565	0.199077	ns
Multimodal Fusion 1 vs Multimodal Fusion 2	QED	8.198715	1.49E-15	****
Multimodal Fusion 1 vs Multimodal Fusion 2	SAscore	-7.84391	2.11E-14	****

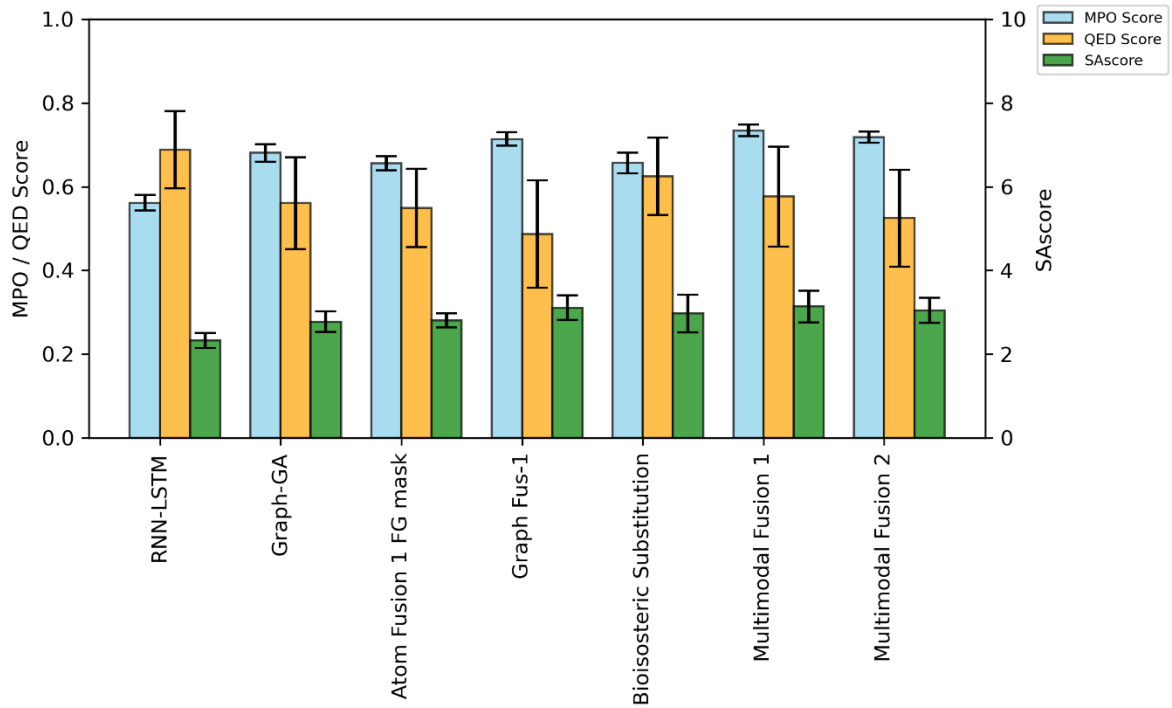


Figure 0-10. The Average Sitagliptin MPO Task, SAscore, and QED Scores of the Molecules Generated by the RNN-LSTM Augmented with the Multimodal-Fusion 1 and 2 Operators Compared to Previous Operators and Baselines. The results shown are averaged across three repeats. The left y-axis represents the scale of values for the MPO and QED scores. The right y-axis represents the scale of values for the SAscores.

Table 0-18. Welch's t-test Results for the Pairwise Comparison of the baselines (RNN-LSTM and Graph-GA) and the augmented RNN-LSTM (Atom-fusion 3 with FG mask, Graph—fusion 1, Bioisosteric Substitution, Multimodal-fusion 1, Multimodal-fusion 2) when tasked on the Zaleplon MPO. The significance indicates: **** (p <= 0.0001), *** (p <= 0.001), ** (p <= 0.01), * (p <= 0.05), and ns (p > 0.05).

Comparison	Score	t-statistic	p-value	Significance
RNN-LSTM vs Graph-GA	MPO	-65.3377	1.16E-229	****
RNN-LSTM vs Graph-GA	QED	13.62829	2.27E-35	****
RNN-LSTM vs Graph-GA	SAscore	-22.4982	2.63E-72	****
RNN-LSTM vs Atom Fusion 1 FG mask	MPO	-61.4503	1.11E-240	****
RNN-LSTM vs Atom Fusion 1 FG mask	QED	17.40177	4.62E-54	****
RNN-LSTM vs Atom Fusion 1 FG mask	SAscore	-32.6682	1.54E-128	****
RNN-LSTM vs Graph Fus-1	MPO	-100.666	0	****
RNN-LSTM vs Graph Fus-1	QED	21.17848	7.51E-72	****
RNN-LSTM vs Graph Fus-1	SAscore	-37.7066	1.01E-143	****
RNN-LSTM vs Bioisosteric Substitution	MPO	-47.6074	3.42E-169	****
RNN-LSTM vs Bioisosteric Substitution	QED	7.561302	2.13E-13	****
RNN-LSTM vs Bioisosteric Substitution	SAscore	-20.1692	1.95E-56	****
RNN-LSTM vs Multimodal Fusion 1	MPO	-121.433	0	****
RNN-LSTM vs Multimodal Fusion 1	QED	12.2645	1.43E-30	****
RNN-LSTM vs Multimodal Fusion 1	SAscore	-32.1847	1.80E-113	****
RNN-LSTM vs Multimodal Fusion 2	MPO	-110.519	0	****
RNN-LSTM vs Multimodal Fusion 2	QED	17.76196	8.91E-55	****
RNN-LSTM vs Multimodal Fusion 2	SAscore	-33.0072	6.26E-118	****
Graph-GA vs Atom Fusion 1 FG mask	MPO	14.46323	1.29E-38	****
Graph-GA vs Atom Fusion 1 FG mask	QED	1.262058	0.207616	ns
Graph-GA vs Atom Fusion 1 FG mask	SAscore	-1.6116	0.107927	ns
Graph-GA vs Graph Fus-1	MPO	-19.5663	1.58E-60	****
Graph-GA vs Graph Fus-1	QED	7.002085	8.28E-12	****
Graph-GA vs Graph Fus-1	SAscore	-13.6542	2.98E-36	****
Graph-GA vs Bioisosteric Substitution	MPO	11.13368	1.81E-25	****
Graph-GA vs Bioisosteric Substitution	QED	-6.58408	1.35E-10	****
Graph-GA vs Bioisosteric Substitution	SAscore	-5.63833	3.58E-08	****
Graph-GA vs Multimodal Fusion 1	MPO	-32.9023	1.06E-110	****
Graph-GA vs Multimodal Fusion 1	QED	-1.50822	0.132151	ns
Graph-GA vs Multimodal Fusion 1	SAscore	-12.8339	1.19E-32	****
Graph-GA vs Multimodal Fusion 2	MPO	-23.3445	6.84E-74	****

Graph-GA vs Multimodal Fusion 2	QED	3.484289	0.00054	***
Graph-GA vs Multimodal Fusion 2	SAscore	-10.6409	7.54E-24	****
Atom Fusion 1 FG mask vs Graph Fus-1	MPO	-41.7869	2.69E-173	****
Atom Fusion 1 FG mask vs Graph Fus-1	QED	6.625389	8.80E-11	****
Atom Fusion 1 FG mask vs Graph Fus-1	SAscore	-14.9111	5.83E-41	****
Atom Fusion 1 FG mask vs Bioisosteric Substitution	MPO	-0.53647	0.591952	ns
Atom Fusion 1 FG mask vs Bioisosteric Substitution	QED	-9.00881	5.15E-18	****
Atom Fusion 1 FG mask vs Bioisosteric Substitution	SAscore	-5.14614	5.16E-07	****
Atom Fusion 1 FG mask vs Multimodal Fusion 1	MPO	-60.5729	4.75E-241	****
Atom Fusion 1 FG mask vs Multimodal Fusion 1	QED	-3.00338	0.002797	**
Atom Fusion 1 FG mask vs Multimodal Fusion 1	SAscore	-13.3699	1.09E-33	****
Atom Fusion 1 FG mask vs Multimodal Fusion 2	MPO	-48.5846	9.05E-195	****
Atom Fusion 1 FG mask vs Multimodal Fusion 2	QED	2.660249	0.008065	**
Atom Fusion 1 FG mask vs Multimodal Fusion 2	SAscore	-11.1592	2.96E-25	****
Graph Fus-1 vs Bioisosteric Substitution	MPO	30.26965	4.04E-102	****
Graph Fus-1 vs Bioisosteric Substitution	QED	-14.031	6.60E-38	****
Graph Fus-1 vs Bioisosteric Substitution	SAscore	3.959731	9.03E-05	****
Graph Fus-1 vs Multimodal Fusion 1	MPO	-15.7395	2.77E-46	****
Graph Fus-1 vs Multimodal Fusion 1	QED	-8.61736	7.09E-17	****
Graph Fus-1 vs Multimodal Fusion 1	SAscore	-1.07745	0.281772	ns
Graph Fus-1 vs Multimodal Fusion 2	MPO	-3.55346	0.000414	***
Graph Fus-1 vs Multimodal Fusion 2	QED	-3.65859	0.000279	***
Graph Fus-1 vs Multimodal Fusion 2	SAscore	2.480203	0.013441	*
Bioisosteric Substitution vs Multimodal Fusion 1	MPO	-42.5323	3.52E-136	****
Bioisosteric Substitution vs Multimodal Fusion 1	QED	5.094961	4.95E-07	****
Bioisosteric Substitution vs Multimodal Fusion 1	SAscore	-4.47994	9.57E-06	****

Bioisosteric Substitution vs Multimodal Fusion 2	MPO	-33.9913	1.15E-109	****
Bioisosteric Substitution vs Multimodal Fusion 2	QED	10.46553	3.42E-23	****
Bioisosteric Substitution vs Multimodal Fusion 2	SAscore	-2.08836	0.037437	*
Multimodal Fusion 1 vs Multimodal Fusion 2	MPO	13.36382	2.41E-35	****
Multimodal Fusion 1 vs Multimodal Fusion 2	QED	5.081558	5.19E-07	****
Multimodal Fusion 1 vs Multimodal Fusion 2	SAscore	3.213205	0.001393	**

Finetuning Dataset Filtering

Baselines - Sitagliptin

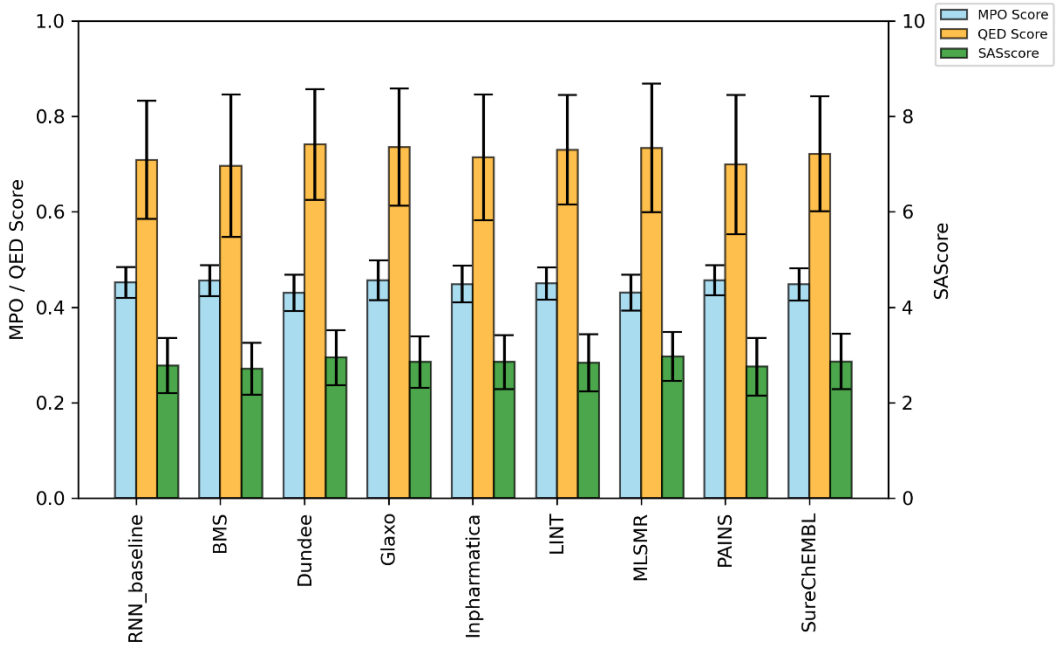


Figure 0-11. The Average Sitagliptin MPO Task, SAscore, and QED Scores of the Molecules Generated by the RNN Baseline with and without Filtering. The results shown are averaged across three repeats. The left y-axis represents the scale of values for the MPO and QED scores. The right y-axis represents the scale of values for the SAscores.

Table 0-19. Welch's t-test Results for the Pairwise Comparison of the RNN-LSTM Baseline with and without Filtering when Tasked on the Sitagliptin MPO. The significance indicates: **** (p <= 0.0001), *** (p <= 0.001), ** (p <= 0.01), * (p <= 0.05), and ns (p > 0.05).

Comparison	Score	t-statistic	p-value	Significance
RNN-LSTM vs BMS	MPO	-0.49351	0.621931	ns
RNN-LSTM vs BMS	QED	-0.39905	0.690074	ns
RNN-LSTM vs BMS	SAscore	-0.26754	0.789195	ns
RNN-LSTM vs Dundee	MPO	5.059962	6.80E-07	****
RNN-LSTM vs Dundee	QED	-1.69299	0.0913	ns
RNN-LSTM vs Dundee	SAscore	-1.76698	0.078061	ns
RNN-LSTM vs Glaxo	MPO	1.219754	0.223308	ns
RNN-LSTM vs Glaxo	QED	-1.18492	0.236782	ns
RNN-LSTM vs Glaxo	SAscore	-1.72944	0.084533	ns
RNN-LSTM vs Inpharmatica	MPO	-0.86545	0.387354	ns
RNN-LSTM vs Inpharmatica	QED	-2.4326	0.01545	*
RNN-LSTM vs Inpharmatica	SAscore	-1.34148	0.180545	ns
RNN-LSTM vs LINT	MPO	1.807389	0.071495	ns
RNN-LSTM vs LINT	QED	-1.12939	0.259446	ns
RNN-LSTM vs LINT	SAscore	0.606236	0.544722	ns
RNN-LSTM vs MLSMR	MPO	6.433725	4.39E-10	****
RNN-LSTM vs MLSMR	QED	-2.14148	0.03292	*
RNN-LSTM vs MLSMR	SAscore	-2.22799	0.026516	*
RNN-LSTM vs PAINS	MPO	0.431302	0.666487	ns
RNN-LSTM vs PAINS	QED	1.339427	0.181215	ns
RNN-LSTM vs PAINS	SAscore	0.379678	0.70439	ns
RNN-LSTM vs SureChEMBL	MPO	0.681402	0.496015	ns
RNN-LSTM vs SureChEMBL	QED	-1.69843	0.090221	ns
RNN-LSTM vs SureChEMBL	SAscore	-0.51965	0.603597	ns
BMS vs Dundee	MPO	5.243348	2.66E-07	****
BMS vs Dundee	QED	-1.25495	0.210277	ns
BMS vs Dundee	SAscore	-1.47629	0.140706	ns
BMS vs Glaxo	MPO	1.644256	0.100951	ns
BMS vs Glaxo	QED	-0.76069	0.447303	ns
BMS vs Glaxo	SAscore	-1.42792	0.154115	ns
BMS vs Inpharmatica	MPO	-0.39371	0.694012	ns
BMS vs Inpharmatica	QED	-1.97158	0.049371	*
BMS vs Inpharmatica	SAscore	-1.04822	0.295176	ns
BMS vs LINT	MPO	2.182756	0.029661	*
BMS vs LINT	QED	-0.71567	0.474627	ns
BMS vs LINT	SAscore	0.872575	0.383442	ns
BMS vs MLSMR	MPO	6.542156	2.16E-10	****
BMS vs MLSMR	QED	-1.70124	0.089764	ns
BMS vs MLSMR	SAscore	-1.92568	0.054939	ns
BMS vs PAINS	MPO	0.874593	0.382322	ns
BMS vs PAINS	QED	1.696629	0.090554	ns

BMS vs PAINS	SAscore	0.642985	0.520604	ns
BMS vs SureChEMBL	MPO	1.128209	0.259915	ns
BMS vs SureChEMBL	QED	-1.26099	0.208055	ns
BMS vs SureChEMBL	SAscore	-0.24522	0.806414	ns
Dundee vs Glaxo	MPO	-4.10919	4.99E-05	****
Dundee vs Glaxo	QED	0.501954	0.615998	ns
Dundee vs Glaxo	SAscore	0.101172	0.919469	ns
Dundee vs Inpharmatica	MPO	-5.25384	2.50E-07	****
Dundee vs Inpharmatica	QED	-0.72546	0.468626	ns
Dundee vs Inpharmatica	SAscore	0.47202	0.637189	ns
Dundee vs LINT	MPO	-3.40796	0.000731	***
Dundee vs LINT	QED	0.519164	0.603962	ns
Dundee vs LINT	SAscore	2.423289	0.015875	*
Dundee vs MLSMR	MPO	1.157165	0.24802	ns
Dundee vs MLSMR	QED	-0.48288	0.629497	ns
Dundee vs MLSMR	SAscore	-0.42939	0.667914	ns
Dundee vs PAINS	MPO	-4.4353	1.21E-05	****
Dundee vs PAINS	QED	2.953507	0.003342	**
Dundee vs PAINS	SAscore	2.15317	0.031943	*
Dundee vs SureChEMBL	MPO	-4.41891	1.32E-05	****
Dundee vs SureChEMBL	QED	-0.00831	0.99337	ns
Dundee vs SureChEMBL	SAscore	1.246856	0.213221	ns
Glaxo vs Inpharmatica	MPO	-1.92166	0.055443	ns
Glaxo vs Inpharmatica	QED	-1.22882	0.219884	ns
Glaxo vs Inpharmatica	SAscore	0.385551	0.700038	ns
Glaxo vs LINT	MPO	0.657604	0.511195	ns
Glaxo vs LINT	QED	0.02914	0.976768	ns
Glaxo vs LINT	SAscore	2.412927	0.016298	*
Glaxo vs MLSMR	MPO	5.501835	7.73E-08	****
Glaxo vs MLSMR	QED	-0.97226	0.331587	ns
Glaxo vs MLSMR	SAscore	-0.54779	0.584187	ns
Glaxo vs PAINS	MPO	-0.70114	0.483639	ns
Glaxo vs PAINS	QED	2.466755	0.01407	*
Glaxo vs PAINS	SAscore	2.130377	0.033759	*
Glaxo vs SureChEMBL	MPO	-0.50588	0.613222	ns
Glaxo vs SureChEMBL	QED	-0.50934	0.610798	ns
Glaxo vs SureChEMBL	SAscore	1.190123	0.23471	ns
Inpharmatica vs LINT	MPO	2.411672	0.016367	*
Inpharmatica vs LINT	QED	1.227224	0.220508	ns
Inpharmatica vs LINT	SAscore	2.006033	0.04555	*
Inpharmatica vs MLSMR	MPO	6.453033	3.54E-10	****
Inpharmatica vs MLSMR	QED	0.221338	0.824959	ns
Inpharmatica vs MLSMR	SAscore	-0.9213	0.35752	ns
Inpharmatica vs PAINS	MPO	1.202805	0.229782	ns

Inpharmatica vs PAINS	QED	3.661991	0.000286	***
Inpharmatica vs PAINS	SAscore	1.736935	0.083169	ns
Inpharmatica vs SureChEMBL	MPO	1.445125	0.149253	ns
Inpharmatica vs SureChEMBL	QED	0.715651	0.474623	ns
Inpharmatica vs SureChEMBL	SAscore	0.807433	0.419894	ns
LINT vs MLSMR	MPO	4.737959	3.21E-06	****
LINT vs MLSMR	QED	-0.97804	0.328736	ns
LINT vs MLSMR	SAscore	-2.91125	0.003838	**
LINT vs PAINS	MPO	-1.27694	0.20239	ns
LINT vs PAINS	QED	2.391435	0.017263	*
LINT vs PAINS	SAscore	-0.2177	0.827774	ns
LINT vs SureChEMBL	MPO	-1.11784	0.264325	ns
LINT vs SureChEMBL	QED	-0.52636	0.598941	ns
LINT vs SureChEMBL	SAscore	-1.14081	0.25465	ns
MLSMR vs PAINS	MPO	-5.72734	2.20E-08	****
MLSMR vs PAINS	QED	3.361009	0.00086	***
MLSMR vs PAINS	SAscore	2.623252	0.009084	**
MLSMR vs SureChEMBL	MPO	-5.77022	1.79E-08	****
MLSMR vs SureChEMBL	QED	0.47387	0.635881	ns
MLSMR vs SureChEMBL	SAscore	1.698209	0.090332	ns
PAINS vs SureChEMBL	MPO	0.214342	0.830389	ns
PAINS vs SureChEMBL	QED	-2.95709	0.003296	**
PAINS vs SureChEMBL	SAscore	-0.90119	0.368025	ns

Baselines - Zaleplon

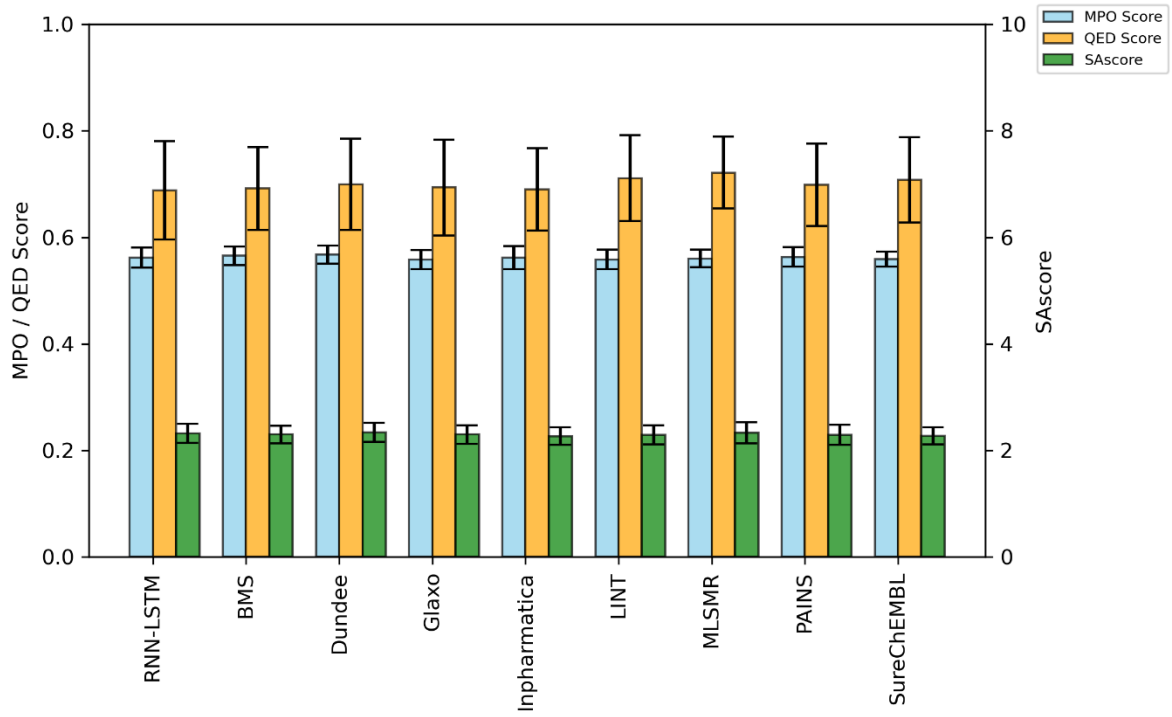


Figure 0-12. The Average Zaleplon MPO Task, SAscore, and QED Scores of the Molecules Generated by the RNN Baseline with and without Filtering. The results shown are averaged across three repeats. The left y-axis represents the scale of values for the MPO and QED scores. The right y-axis represents the scale of values for the SAscores.

Table 0-20. Welch's t-test Results for the Pairwise Comparison of the RNN-LSTM Baseline with and without Filtering when Tasked on the Zaleplon MPO. The significance indicates: **** (p <= 0.0001), *** (p <= 0.001), ** (p <= 0.01), * (p <= 0.05), and ns (p > 0.05).

Comparison	Score	t-statistic	p-value	Significance
RNN-LSTM vs BMS	MPO	-2.40645	0.016457	*
RNN-LSTM vs BMS	QED	-0.46404	0.64282	ns
RNN-LSTM vs BMS	SAscore	1.539384	0.124322	ns
RNN-LSTM vs Dundee	MPO	-3.74655	0.0002	***
RNN-LSTM vs Dundee	QED	-1.41306	0.158264	ns
RNN-LSTM vs Dundee	SAscore	-0.81005	0.418306	ns
RNN-LSTM vs Glaxo	MPO	2.322206	0.020604	*
RNN-LSTM vs Glaxo	QED	-0.68186	0.49563	ns
RNN-LSTM vs Glaxo	SAscore	1.730962	0.084045	ns
RNN-LSTM vs Inpharmatica	MPO	-0.23489	0.814381	ns
RNN-LSTM vs Inpharmatica	QED	-0.22346	0.823267	ns
RNN-LSTM vs Inpharmatica	SAscore	3.818725	0.00015	***
RNN-LSTM vs LINT	MPO	2.081087	0.037913	*
RNN-LSTM vs LINT	QED	-3.02324	0.002626	**
RNN-LSTM vs LINT	SAscore	1.974345	0.048869	*
RNN-LSTM vs MLSMR	MPO	0.854553	0.393196	ns
RNN-LSTM vs MLSMR	QED	-4.64997	4.30E-06	****
RNN-LSTM vs MLSMR	SAscore	-0.46516	0.642011	ns
RNN-LSTM vs PAINS	MPO	-0.86341	0.388308	ns
RNN-LSTM vs PAINS	QED	-1.35656	0.175524	ns
RNN-LSTM vs PAINS	SAscore	1.984751	0.047693	*
RNN-LSTM vs SureChEMBL	MPO	1.760725	0.078914	ns
RNN-LSTM vs SureChEMBL	QED	-2.59222	0.009812	**
RNN-LSTM vs SureChEMBL	SAscore	3.401923	0.000722	***
BMS vs Dundee	MPO	-1.39825	0.162668	ns
BMS vs Dundee	QED	-1.05898	0.290144	ns
BMS vs Dundee	SAscore	-2.33083	0.020176	*
BMS vs Glaxo	MPO	4.889292	1.35E-06	****
BMS vs Glaxo	QED	-0.26555	0.790689	ns
BMS vs Glaxo	SAscore	0.228514	0.819335	ns
BMS vs Inpharmatica	MPO	2.003616	0.045624	*
BMS vs Inpharmatica	QED	0.271573	0.786055	ns
BMS vs Inpharmatica	SAscore	2.333003	0.02002	*
BMS vs LINT	MPO	4.59235	5.50E-06	****
BMS vs LINT	QED	-2.80275	0.005256	**
BMS vs LINT	SAscore	0.523358	0.60095	ns
BMS vs MLSMR	MPO	3.41257	0.000694	***
BMS vs MLSMR	QED	-4.61015	5.10E-06	****
BMS vs MLSMR	SAscore	-1.94207	0.052685	ns
BMS vs PAINS	MPO	1.561522	0.119006	ns
BMS vs PAINS	QED	-0.97618	0.329428	ns

BMS vs PAINS	SAscore	0.542498	0.587709	ns
BMS vs SureChEMBL	MPO	4.536609	7.18E-06	****
BMS vs SureChEMBL	QED	-2.33221	0.02008	*
BMS vs SureChEMBL	SAscore	1.920983	0.055292	ns
Dundee vs Glaxo	MPO	6.259096	8.35E-10	****
Dundee vs Glaxo	QED	0.746969	0.455434	ns
Dundee vs Glaxo	SAscore	2.503823	0.01261	*
Dundee vs Inpharmatica	MPO	3.257806	0.001198	**
Dundee vs Inpharmatica	QED	1.325088	0.185771	ns
Dundee vs Inpharmatica	SAscore	4.555787	6.63E-06	****
Dundee vs LINT	MPO	5.939921	5.37E-09	****
Dundee vs LINT	QED	-1.55585	0.120397	ns
Dundee vs LINT	SAscore	2.721656	0.006725	**
Dundee vs MLSMR	MPO	4.812506	1.99E-06	****
Dundee vs MLSMR	QED	-3.10226	0.002042	**
Dundee vs MLSMR	SAscore	0.320536	0.748697	ns
Dundee vs PAINS	MPO	2.931889	0.003524	**
Dundee vs PAINS	QED	0.162074	0.871316	ns
Dundee vs PAINS	SAscore	2.728093	0.006596	**
Dundee vs SureChEMBL	MPO	6.023506	3.49E-09	****
Dundee vs SureChEMBL	QED	-1.12672	0.260424	ns
Dundee vs SureChEMBL	SAscore	4.143248	4.06E-05	****
Glaxo vs Inpharmatica	MPO	-2.40699	0.016426	*
Glaxo vs Inpharmatica	QED	0.521594	0.602173	ns
Glaxo vs Inpharmatica	SAscore	2.048922	0.040956	*
Glaxo vs LINT	MPO	-0.2081	0.835229	ns
Glaxo vs LINT	QED	-2.35234	0.019026	*
Glaxo vs LINT	SAscore	0.296148	0.767234	ns
Glaxo vs MLSMR	MPO	-1.55573	0.120375	ns
Glaxo vs MLSMR	QED	-3.96872	8.30E-05	****
Glaxo vs MLSMR	SAscore	-2.11889	0.034578	*
Glaxo vs PAINS	MPO	-3.25974	0.001187	**
Glaxo vs PAINS	QED	-0.64067	0.522017	ns
Glaxo vs PAINS	SAscore	0.316327	0.751879	ns
Glaxo vs SureChEMBL	MPO	-0.78107	0.435126	ns
Glaxo vs SureChEMBL	QED	-1.91657	0.055846	ns
Glaxo vs SureChEMBL	SAscore	1.650569	0.099436	ns
Inpharmatica vs LINT	MPO	2.182875	0.029482	*
Inpharmatica vs LINT	QED	-3.11595	0.001932	**
Inpharmatica vs LINT	SAscore	-1.67553	0.094431	ns
Inpharmatica vs MLSMR	MPO	1.040007	0.29882	ns
Inpharmatica vs MLSMR	QED	-4.99162	8.12E-07	****
Inpharmatica vs MLSMR	SAscore	-4.10773	4.66E-05	****
Inpharmatica vs PAINS	MPO	-0.56808	0.570218	ns

Inpharmatica vs PAINS	QED	-1.26639	0.20592	ns
Inpharmatica vs PAINS	SAscore	-1.64198	0.101195	ns
Inpharmatica vs SureChEMBL	MPO	1.882428	0.06038	ns
Inpharmatica vs SureChEMBL	QED	-2.63547	0.008653	**
Inpharmatica vs SureChEMBL	SAscore	-0.38486	0.700501	ns
LINT vs MLSMR	MPO	-1.31508	0.189066	ns
LINT vs MLSMR	QED	-1.53335	0.125817	ns
LINT vs MLSMR	SAscore	-2.34342	0.019486	*
LINT vs PAINS	MPO	-2.9967	0.002858	**
LINT vs PAINS	QED	1.859566	0.063508	ns
LINT vs PAINS	SAscore	0.021322	0.982997	ns
LINT vs SureChEMBL	MPO	-0.53855	0.590441	ns
LINT vs SureChEMBL	QED	0.446171	0.655663	ns
LINT vs SureChEMBL	SAscore	1.295712	0.195665	ns
MLSMR vs PAINS	MPO	-1.77774	0.07603	ns
MLSMR vs PAINS	QED	3.597662	0.000352	***
MLSMR vs PAINS	SAscore	2.351949	0.019047	*
MLSMR vs SureChEMBL	MPO	0.901673	0.367665	ns
MLSMR vs SureChEMBL	QED	2.005841	0.045424	*
MLSMR vs SureChEMBL	SAscore	3.71144	0.00023	***
PAINS vs SureChEMBL	MPO	2.768875	0.005835	**
PAINS vs SureChEMBL	QED	-1.39203	0.164519	ns
PAINS vs SureChEMBL	SAscore	1.265157	0.206394	ns

RNN-LSTM with Augmentation at the Subset Loop = 5 Point – Sitagliptin MPO

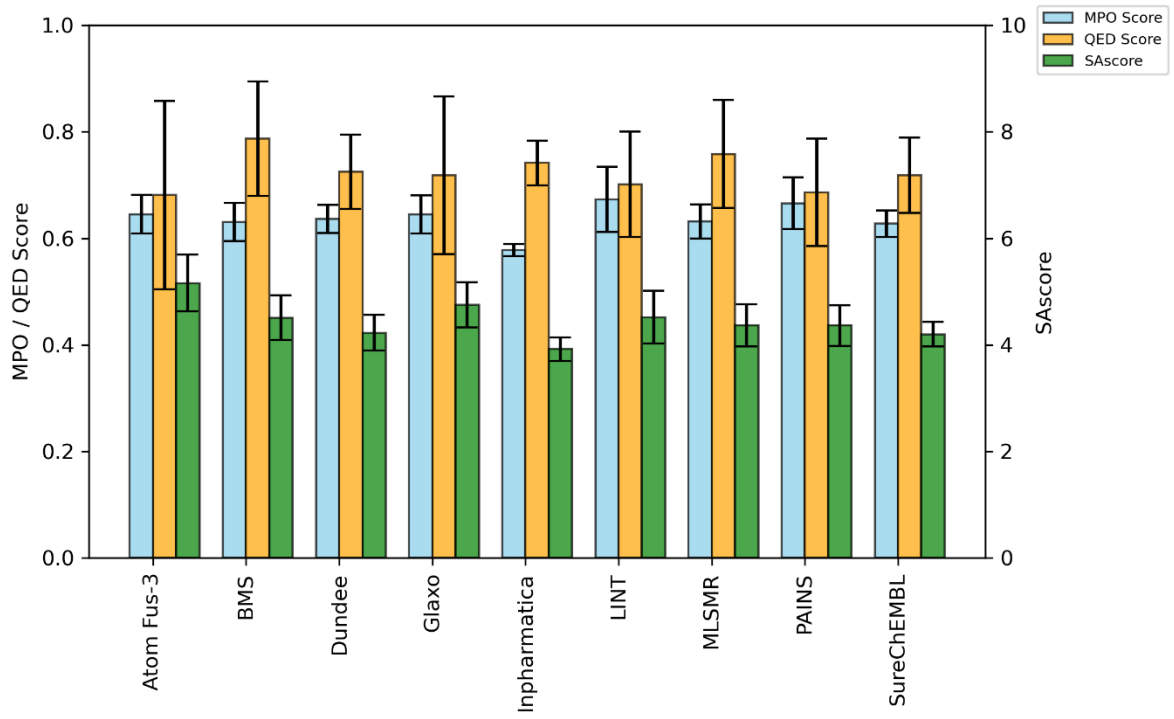


Figure 0-13. The Average Sitagliptin MPO Task, SAscore, and QED Scores of the Molecules Generated by the RNN-LSTM Augmented using Atom-fusion 3 with and without Filtering. The results shown are averaged across three repeats. The left y-axis represents the scale of values for the MPO and QED scores. The right y-axis represents the scale of values for the SAscores.

Table 0-21. Welch's t-test Results for the Pairwise Comparison of the RNN-LSTM Augmented using Atom-fusion 3 with and without Filtering when Tasked on the Sitagliptin MPO. The significance indicates: **** (p <= 0.0001), *** (p <= 0.001), ** (p <= 0.01), * (p <= 0.05), and ns (p > 0.05).

Comparison	Score	t-statistic	p-value	Significance
Atom Fus-3 vs BMS	MPO	4.893705	1.28E-06	****
Atom Fus-3 vs BMS	QED	-8.88146	1.22E-17	****
Atom Fus-3 vs BMS	SAscore	16.49758	3.21E-50	****
Atom Fus-3 vs Dundee	MPO	3.504986	0.000494	***
Atom Fus-3 vs Dundee	QED	-4.01438	7.15E-05	****
Atom Fus-3 vs Dundee	SAscore	25.61699	4.77E-93	****
Atom Fus-3 vs Glaxo	MPO	0.228245	0.819534	ns
Atom Fus-3 vs Glaxo	QED	-2.80341	0.005226	**
Atom Fus-3 vs Glaxo	SAscore	10.38678	2.92E-23	****
Atom Fus-3 vs Inpharmatica	MPO	30.31603	2.01E-101	****
Atom Fus-3 vs Inpharmatica	QED	-5.74006	2.10E-08	****
Atom Fus-3 vs Inpharmatica	SAscore	36.57267	1.68E-131	****
Atom Fus-3 vs LINT	MPO	-6.64511	8.37E-11	****
Atom Fus-3 vs LINT	QED	-1.7496	0.080836	ns
Atom Fus-3 vs LINT	SAscore	15.12554	6.64E-44	****
Atom Fus-3 vs MLSMR	MPO	4.842368	1.65E-06	****
Atom Fus-3 vs MLSMR	QED	-6.52803	1.69E-10	****
Atom Fus-3 vs MLSMR	SAscore	20.56846	4.15E-70	****
Atom Fus-3 vs PAINS	MPO	-5.89436	6.57E-09	****
Atom Fus-3 vs PAINS	QED	-0.4555	0.648956	ns
Atom Fus-3 vs PAINS	SAscore	21.02888	3.70E-72	****
Atom Fus-3 vs SureChEMBL	MPO	6.951383	1.08E-11	****
Atom Fus-3 vs SureChEMBL	QED	-3.3718	0.00082	***
Atom Fus-3 vs SureChEMBL	SAscore	28.46617	3.31E-99	****
BMS vs Dundee	MPO	-2.09888	0.036293	*
BMS vs Dundee	QED	8.352299	6.54E-16	****
BMS vs Dundee	SAscore	9.002492	3.48E-18	****
BMS vs Glaxo	MPO	-4.72754	2.84E-06	****
BMS vs Glaxo	QED	6.513475	1.67E-10	****
BMS vs Glaxo	SAscore	-6.92579	1.13E-11	****
BMS vs Inpharmatica	MPO	24.05414	9.31E-77	****
BMS vs Inpharmatica	QED	6.74582	5.47E-11	****
BMS vs Inpharmatica	SAscore	20.82854	2.90E-68	****
BMS vs LINT	MPO	-10.1672	4.49E-22	****
BMS vs LINT	QED	10.03978	5.55E-22	****
BMS vs LINT	SAscore	-0.24915	0.803334	ns
BMS vs MLSMR	MPO	-0.2761	0.782572	ns
BMS vs MLSMR	QED	3.350392	0.00086	***
BMS vs MLSMR	SAscore	4.263048	2.36E-05	****
BMS vs PAINS	MPO	-10.0785	5.14E-22	****
BMS vs PAINS	QED	11.73843	1.01E-28	****

BMS vs PAINS	SAscore	4.469623	9.42E-06	****
BMS vs SureChEMBL	MPO	1.291048	0.197258	ns
BMS vs SureChEMBL	QED	9.176621	1.10E-18	****
BMS vs SureChEMBL	SAscore	11.03789	2.78E-25	****
Dundee vs Glaxo	MPO	-3.30137	0.001025	**
Dundee vs Glaxo	QED	0.713118	0.476163	ns
Dundee vs Glaxo	SAscore	-16.705	3.47E-51	****
Dundee vs Inpharmatica	MPO	34.9388	1.76E-125	****
Dundee vs Inpharmatica	QED	-3.40544	0.000714	***
Dundee vs Inpharmatica	SAscore	12.95856	1.74E-33	****
Dundee vs LINT	MPO	-9.44279	3.31E-19	****
Dundee vs LINT	QED	3.322018	0.000956	***
Dundee vs LINT	SAscore	-8.3282	7.74E-16	****
Dundee vs MLSMR	MPO	1.907942	0.05692	ns
Dundee vs MLSMR	QED	-4.59067	5.59E-06	****
Dundee vs MLSMR	SAscore	-4.52568	7.37E-06	****
Dundee vs PAINS	MPO	-9.35066	4.02E-19	****
Dundee vs PAINS	QED	5.414438	9.40E-08	****
Dundee vs PAINS	SAscore	-4.54157	6.80E-06	****
Dundee vs SureChEMBL	MPO	4.143556	3.93E-05	****
Dundee vs SureChEMBL	QED	1.180824	0.238154	ns
Dundee vs SureChEMBL	SAscore	1.248114	0.212538	ns
Glaxo vs Inpharmatica	MPO	30.71849	2.60E-103	****
Glaxo vs Inpharmatica	QED	-2.59086	0.009969	**
Glaxo vs Inpharmatica	SAscore	29.3131	1.16E-107	****
Glaxo vs LINT	MPO	-6.85133	2.33E-11	****
Glaxo vs LINT	QED	1.622698	0.105255	ns
Glaxo vs LINT	SAscore	6.073467	2.28E-09	****
Glaxo vs MLSMR	MPO	4.670765	3.73E-06	****
Glaxo vs MLSMR	QED	-3.82625	0.000146	***
Glaxo vs MLSMR	SAscore	11.37049	3.40E-27	****
Glaxo vs PAINS	MPO	-6.14237	1.57E-09	****
Glaxo vs PAINS	QED	3.077267	0.002197	**
Glaxo vs PAINS	SAscore	11.74397	9.41E-29	****
Glaxo vs SureChEMBL	MPO	6.806019	2.71E-11	****
Glaxo vs SureChEMBL	QED	0.013513	0.989224	ns
Glaxo vs SureChEMBL	SAscore	19.57929	1.06E-62	****
Inpharmatica vs LINT	MPO	-26.0206	3.06E-80	****
Inpharmatica vs LINT	QED	6.261261	9.79E-10	****
Inpharmatica vs LINT	SAscore	-18.5377	2.65E-56	****
Inpharmatica vs MLSMR	MPO	-26.5573	1.15E-86	****
Inpharmatica vs MLSMR	QED	-2.56149	0.0108	*
Inpharmatica vs MLSMR	SAscore	-16.3764	9.06E-48	****
Inpharmatica vs PAINS	MPO	-30.505	4.80E-98	****

Inpharmatica vs PAINS	QED	8.539928	2.76E-16	****
Inpharmatica vs PAINS	SAscore	-16.8564	1.69E-50	****
Inpharmatica vs SureChEMBL	MPO	-30.8605	4.34E-109	****
Inpharmatica vs SureChEMBL	QED	4.711545	3.23E-06	****
Inpharmatica vs SureChEMBL	SAscore	-14.2677	2.28E-39	****
LINT vs MLSMR	MPO	10.20459	4.30E-22	****
LINT vs MLSMR	QED	-6.79133	2.78E-11	****
LINT vs MLSMR	SAscore	4.128023	4.23E-05	****
LINT vs PAINS	MPO	1.531949	0.126109	ns
LINT vs PAINS	QED	1.833899	0.067177	ns
LINT vs PAINS	SAscore	4.303835	1.99E-05	****
LINT vs SureChEMBL	MPO	11.75237	1.91E-27	****
LINT vs SureChEMBL	QED	-2.32402	0.020503	*
LINT vs SureChEMBL	SAscore	9.952312	4.65E-21	****
MLSMR vs PAINS	MPO	-10.1736	2.85E-22	****
MLSMR vs PAINS	QED	8.549328	1.11E-16	****
MLSMR vs PAINS	SAscore	0.109191	0.913089	ns
MLSMR vs SureChEMBL	MPO	1.699544	0.0898	ns
MLSMR vs SureChEMBL	QED	5.478739	6.75E-08	****
MLSMR vs SureChEMBL	SAscore	6.15386	1.65E-09	****
PAINS vs SureChEMBL	MPO	12.19864	1.10E-29	****
PAINS vs SureChEMBL	QED	-4.40223	1.30E-05	****
PAINS vs SureChEMBL	SAscore	6.252329	8.83E-10	****

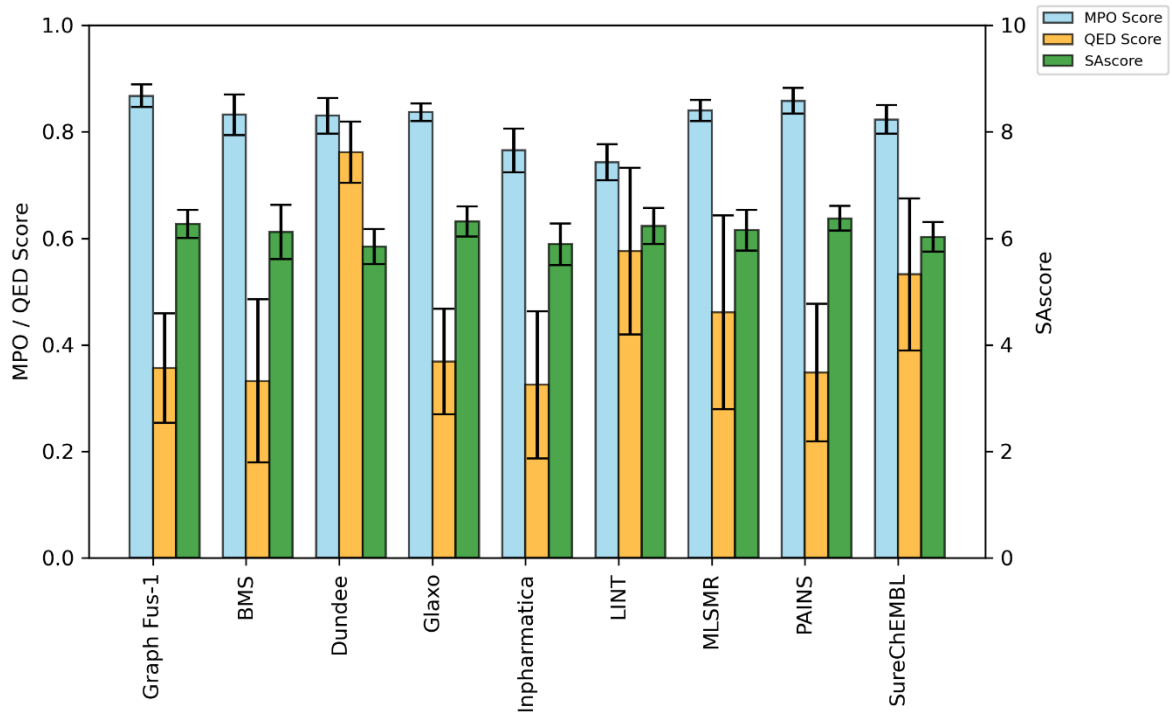


Figure 0-14. The Average Sitagliptin MPO Task, SAscore, and QED Scores of the Molecules Generated by the RNN-LSTM Augmented using Graph-fusion 1 with and without Filtering. The results shown are averaged across three repeats. The left y-axis represents the scale of values for the MPO and QED scores. The right y-axis represents the scale of values for the SAscores.

Table 0-22. Welch's t-test Results for the Pairwise Comparison of the RNN-LSTM Augmented using Graph-fusion 1 with and without Filtering when Tasked on the Sitagliptin MPO. The significance indicates: **** (p <= 0.0001), *** (p <= 0.001), ** (p <= 0.01), * (p <= 0.05), and ns (p > 0.05).

Comparison	Score	t-statistic	p-value	Significance
Graph Fus-1 vs BMS	MPO	14.06592	9.66E-38	****
Graph Fus-1 vs BMS	QED	2.243851	0.025261	*
Graph Fus-1 vs BMS	SAscore	4.562421	6.55E-06	****
Graph Fus-1 vs Dundee	MPO	16.32274	1.92E-48	****
Graph Fus-1 vs Dundee	QED	-59.8049	6.93E-222	****
Graph Fus-1 vs Dundee	SAscore	17.5313	2.53E-55	****
Graph Fus-1 vs Glaxo	MPO	19.7466	2.30E-66	****
Graph Fus-1 vs Glaxo	QED	-1.51824	0.129482	ns
Graph Fus-1 vs Glaxo	SAscore	-2.14943	0.032003	*
Graph Fus-1 vs Inpharmatica	MPO	38.49119	2.57E-144	****
Graph Fus-1 vs Inpharmatica	QED	3.128038	0.001853	**
Graph Fus-1 vs Inpharmatica	SAscore	13.9726	7.13E-38	****
Graph Fus-1 vs LINT	MPO	54.31049	6.58E-213	****
Graph Fus-1 vs LINT	QED	-20.3395	5.94E-68	****
Graph Fus-1 vs LINT	SAscore	1.664361	0.096599	ns
Graph Fus-1 vs MLSMR	MPO	16.57651	5.22E-51	****
Graph Fus-1 vs MLSMR	QED	-8.70865	5.22E-17	****
Graph Fus-1 vs MLSMR	SAscore	4.318858	1.87E-05	****
Graph Fus-1 vs PAINS	MPO	4.898071	1.25E-06	****
Graph Fus-1 vs PAINS	QED	0.849015	0.39623	ns
Graph Fus-1 vs PAINS	SAscore	-5.33021	1.40E-07	****
Graph Fus-1 vs SureChEMBL	MPO	22.29673	1.54E-79	****
Graph Fus-1 vs SureChEMBL	QED	-17.3676	4.72E-54	****
Graph Fus-1 vs SureChEMBL	SAscore	10.95493	1.45E-25	****
BMS vs Dundee	MPO	0.650188	0.515825	ns
BMS vs Dundee	QED	-45.3867	8.95E-156	****
BMS vs Dundee	SAscore	7.781025	4.03E-14	****
BMS vs Glaxo	MPO	-1.96816	0.049727	*
BMS vs Glaxo	QED	-3.45023	0.000606	***
BMS vs Glaxo	SAscore	-5.90059	7.00E-09	****
BMS vs Inpharmatica	MPO	20.79584	1.33E-72	****
BMS vs Inpharmatica	QED	0.598288	0.549877	ns
BMS vs Inpharmatica	SAscore	6.174615	1.27E-09	****
BMS vs LINT	MPO	30.48289	3.19E-123	****
BMS vs LINT	QED	-19.2461	1.34E-64	****
BMS vs LINT	SAscore	-3.10752	0.00199	**
BMS vs MLSMR	MPO	-3.12005	0.001925	**
BMS vs MLSMR	QED	-9.37824	1.48E-19	****
BMS vs MLSMR	SAscore	-0.96738	0.333774	ns
BMS vs PAINS	MPO	-10.0505	8.39E-22	****
BMS vs PAINS	QED	-1.36612	0.172429	ns

BMS vs PAINS	SAscore	-7.97019	1.54E-14	****
BMS vs SureChEMBL	MPO	3.278725	0.00111	**
BMS vs SureChEMBL	QED	-16.5337	8.33E-51	****
BMS vs SureChEMBL	SAscore	2.69257	0.007347	**
Dundee vs Glaxo	MPO	-3.07551	0.002233	**
Dundee vs Glaxo	QED	59.36859	5.08E-223	****
Dundee vs Glaxo	SAscore	-18.9387	1.07E-62	****
Dundee vs Inpharmatica	MPO	21.37978	4.64E-75	****
Dundee vs Inpharmatica	QED	50.59501	7.62E-176	****
Dundee vs Inpharmatica	SAscore	-1.44129	0.150045	ns
Dundee vs LINT	MPO	31.97115	1.56E-131	****
Dundee vs LINT	QED	19.29391	3.12E-58	****
Dundee vs LINT	SAscore	-14.0576	5.52E-39	****
Dundee vs MLSMR	MPO	-4.29929	2.07E-05	****
Dundee vs MLSMR	QED	27.31867	1.34E-89	****
Dundee vs MLSMR	SAscore	-10.5977	4.01E-24	****
Dundee vs PAINS	MPO	-11.7816	1.03E-28	****
Dundee vs PAINS	QED	50.66795	2.78E-179	****
Dundee vs PAINS	SAscore	-22.9262	1.42E-81	****
Dundee vs SureChEMBL	MPO	2.796275	0.005343	**
Dundee vs SureChEMBL	QED	25.82084	1.05E-86	****
Dundee vs SureChEMBL	SAscore	-7.31936	8.30E-13	****
Glaxo vs Inpharmatica	MPO	28.17847	1.90E-96	****
Glaxo vs Inpharmatica	QED	4.436969	1.11E-05	****
Glaxo vs Inpharmatica	SAscore	15.35629	2.11E-44	****
Glaxo vs LINT	MPO	43.4817	1.78E-160	****
Glaxo vs LINT	QED	-19.3617	6.69E-63	****
Glaxo vs LINT	SAscore	3.487224	0.000525	***
Glaxo vs MLSMR	MPO	-2.02489	0.043335	*
Glaxo vs MLSMR	QED	-7.72829	6.84E-14	****
Glaxo vs MLSMR	SAscore	5.949192	4.81E-09	****
Glaxo vs PAINS	MPO	-12.6438	3.54E-32	****
Glaxo vs PAINS	QED	2.18851	0.029044	*
Glaxo vs PAINS	SAscore	-2.84931	0.004538	**
Glaxo vs SureChEMBL	MPO	7.412846	5.39E-13	****
Glaxo vs SureChEMBL	QED	-16.3067	8.47E-49	****
Glaxo vs SureChEMBL	SAscore	12.6605	1.03E-32	****
Inpharmatica vs LINT	MPO	7.287179	1.05E-12	****
Inpharmatica vs LINT	QED	-20.812	1.62E-72	****
Inpharmatica vs LINT	SAscore	-11.3298	4.88E-27	****
Inpharmatica vs MLSMR	MPO	-28.5438	2.96E-101	****
Inpharmatica vs MLSMR	QED	-10.3181	5.82E-23	****
Inpharmatica vs MLSMR	SAscore	-8.37589	3.92E-16	****
Inpharmatica vs PAINS	MPO	-33.9147	2.06E-130	****

Inpharmatica vs PAINS	QED	-2.10324	0.035864	*
Inpharmatica vs PAINS	SAscore	-18.5055	3.81E-58	****
Inpharmatica vs SureChEMBL	MPO	-20.5822	3.19E-69	****
Inpharmatica vs SureChEMBL	QED	-18.0784	1.35E-58	****
Inpharmatica vs SureChEMBL	SAscore	-4.99817	7.84E-07	****
LINT vs MLSMR	MPO	-43.1748	1.26E-167	****
LINT vs MLSMR	QED	8.291311	7.71E-16	****
LINT vs MLSMR	SAscore	2.51924	0.012024	*
LINT vs PAINS	MPO	-48.1664	1.21E-198	****
LINT vs PAINS	QED	19.43749	4.07E-65	****
LINT vs PAINS	SAscore	-6.25512	8.23E-10	****
LINT vs SureChEMBL	MPO	-32.3445	3.37E-131	****
LINT vs SureChEMBL	QED	3.561688	0.000398	***
LINT vs SureChEMBL	SAscore	7.906554	1.35E-14	****
MLSMR vs PAINS	MPO	-10.2405	1.04E-22	****
MLSMR vs PAINS	QED	8.775683	2.24E-17	****
MLSMR vs PAINS	SAscore	-8.61892	9.36E-17	****
MLSMR vs SureChEMBL	MPO	8.585303	9.47E-17	****
MLSMR vs SureChEMBL	QED	-5.34173	1.34E-07	****
MLSMR vs SureChEMBL	SAscore	4.611518	4.98E-06	****
PAINS vs SureChEMBL	MPO	16.68431	1.64E-51	****
PAINS vs SureChEMBL	QED	-16.5764	5.59E-51	****
PAINS vs SureChEMBL	SAscore	16.65609	3.81E-51	****

RNN-LSTM with Augmentation at the Subset Loop Point - Zaleplon

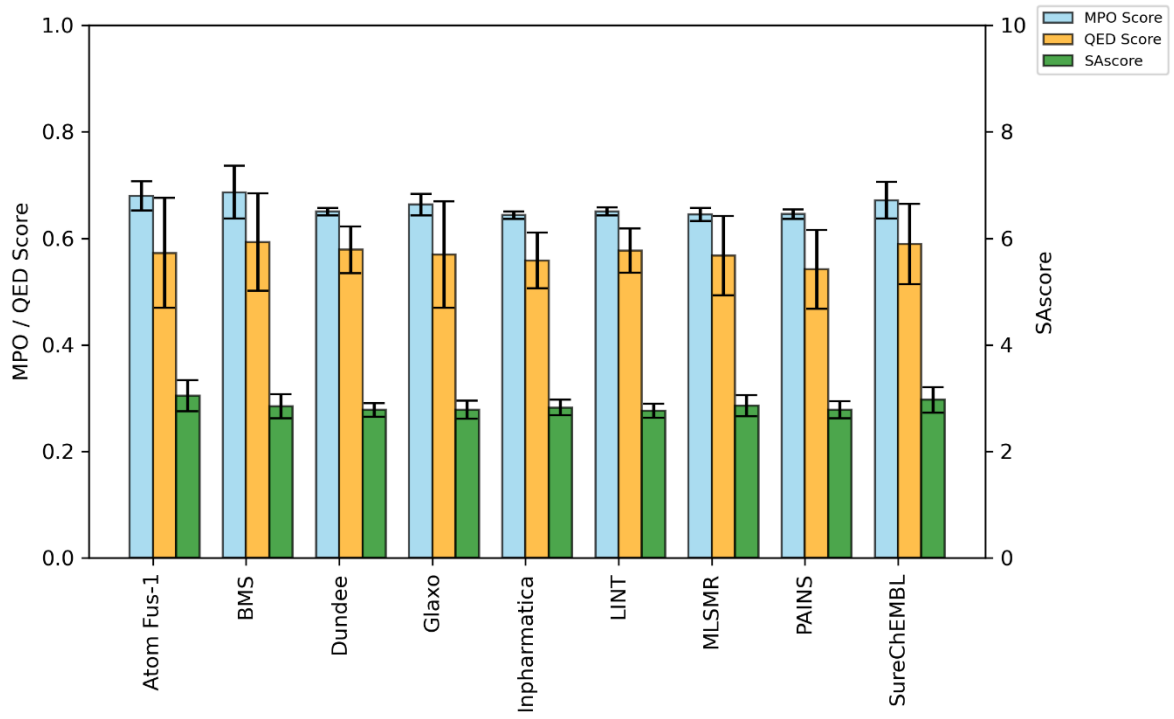


Figure 0-15. The Average Zaleplon MPO Task, SAscore, and QED Scores of the Molecules Generated by the RNN-LSTM Augmented using Atom-fusion 1 with and without Filtering. The results shown are averaged across three repeats. The left y-axis represents the scale of values for the MPO and QED scores. The right y-axis represents the scale of values for the SAscores.

Table 0-23. Welch's t-test Results for the Pairwise Comparison of the RNN-LSTM Augmented using Atom-fusion 1 with and without Filtering when Tasked on the Zaleplon MPO. The significance indicates: **** (p <= 0.0001), *** (p <= 0.001), ** (p <= 0.01), * (p <= 0.05), and ns (p > 0.05).

Comparison	Score	t-statistic	p-value	Significance
Atom Fus-1 vs BMS	MPO	-1.82143	0.069359	ns
Atom Fus-1 vs BMS	QED	-2.44684	0.014729	*
Atom Fus-1 vs BMS	SAscore	8.675606	4.80E-17	****
Atom Fus-1 vs Dundee	MPO	18.13379	3.21E-52	****
Atom Fus-1 vs Dundee	QED	-0.89832	0.369518	ns
Atom Fus-1 vs Dundee	SAscore	13.7731	4.06E-36	****
Atom Fus-1 vs Glaxo	MPO	8.152305	2.50E-15	****
Atom Fus-1 vs Glaxo	QED	0.39555	0.692594	ns
Atom Fus-1 vs Glaxo	SAscore	13.03864	1.42E-33	****
Atom Fus-1 vs Inpharmatica	MPO	21.94354	2.60E-68	****
Atom Fus-1 vs Inpharmatica	QED	1.977382	0.048589	*
Atom Fus-1 vs Inpharmatica	SAscore	10.91835	7.81E-25	****
Atom Fus-1 vs LINT	MPO	17.95086	7.33E-52	****
Atom Fus-1 vs LINT	QED	-0.62509	0.53225	ns
Atom Fus-1 vs LINT	SAscore	14.61535	8.31E-40	****
Atom Fus-1 vs MLSMR	MPO	19.40054	2.90E-61	****
Atom Fus-1 vs MLSMR	QED	0.636943	0.524471	ns
Atom Fus-1 vs MLSMR	SAscore	8.294198	1.12E-15	****
Atom Fus-1 vs PAINS	MPO	20.36977	1.36E-62	****
Atom Fus-1 vs PAINS	QED	4.100115	4.77E-05	****
Atom Fus-1 vs PAINS	SAscore	13.19628	3.94E-34	****
Atom Fus-1 vs SureChEMBL	MPO	3.208758	0.001426	**
Atom Fus-1 vs SureChEMBL	QED	-2.13356	0.033331	*
Atom Fus-1 vs SureChEMBL	SAscore	3.299746	0.001031	**
BMS vs Dundee	MPO	11.40798	1.23E-24	****
BMS vs Dundee	QED	2.191694	0.029038	*
BMS vs Dundee	SAscore	4.101407	4.98E-05	****
BMS vs Glaxo	MPO	6.756747	6.62E-11	****
BMS vs Glaxo	QED	2.7725	0.005772	**
BMS vs Glaxo	SAscore	3.735047	0.000212	***
BMS vs Inpharmatica	MPO	13.45993	1.21E-31	****
BMS vs Inpharmatica	QED	4.870325	1.60E-06	****
BMS vs Inpharmatica	SAscore	1.467065	0.143123	ns
BMS vs LINT	MPO	11.36615	1.56E-24	****
BMS vs LINT	QED	2.507038	0.012627	*
BMS vs LINT	SAscore	5.123555	4.65E-07	****
BMS vs MLSMR	MPO	12.73537	1.09E-29	****
BMS vs MLSMR	QED	3.187871	0.001539	**
BMS vs MLSMR	SAscore	-0.37339	0.709045	ns
BMS vs PAINS	MPO	12.71212	3.86E-29	****
BMS vs PAINS	QED	6.853127	2.28E-11	****

BMS vs PAINS	SAscore	3.797161	0.000167	***
BMS vs SureChEMBL	MPO	3.934104	9.71E-05	****
BMS vs SureChEMBL	QED	0.539016	0.59013	ns
BMS vs SureChEMBL	SAscore	-5.53566	5.08E-08	****
Dundee vs Glaxo	MPO	-9.95429	1.73E-20	****
Dundee vs Glaxo	QED	1.348215	0.178442	ns
Dundee vs Glaxo	SAscore	-0.18466	0.853575	ns
Dundee vs Inpharmatica	MPO	9.592451	1.45E-19	****
Dundee vs Inpharmatica	QED	4.013481	7.41E-05	****
Dundee vs Inpharmatica	SAscore	-3.05817	0.002401	**
Dundee vs LINT	MPO	-0.07371	0.941273	ns
Dundee vs LINT	QED	0.453027	0.650763	ns
Dundee vs LINT	SAscore	1.428201	0.153974	ns
Dundee vs MLSMR	MPO	5.349653	1.85E-07	****
Dundee vs MLSMR	QED	1.780925	0.075976	ns
Dundee vs MLSMR	SAscore	-4.5207	8.81E-06	****
Dundee vs PAINS	MPO	5.968211	4.88E-09	****
Dundee vs PAINS	QED	6.629472	1.07E-10	****
Dundee vs PAINS	SAscore	-0.17736	0.859301	ns
Dundee vs SureChEMBL	MPO	-9.46522	1.67E-18	****
Dundee vs SureChEMBL	QED	-1.81981	0.069534	ns
Dundee vs SureChEMBL	SAscore	-10.4814	9.25E-23	****
Glaxo vs Inpharmatica	MPO	14.67502	6.99E-38	****
Glaxo vs Inpharmatica	QED	1.441078	0.150343	ns
Glaxo vs Inpharmatica	SAscore	-2.65555	0.008233	**
Glaxo vs LINT	MPO	9.792109	4.82E-20	****
Glaxo vs LINT	QED	-1.09224	0.275486	ns
Glaxo vs LINT	SAscore	1.479546	0.139676	ns
Glaxo vs MLSMR	MPO	12.12363	3.08E-29	****
Glaxo vs MLSMR	QED	0.206519	0.836482	ns
Glaxo vs MLSMR	SAscore	-4.1357	4.42E-05	****
Glaxo vs PAINS	MPO	12.81493	4.28E-31	****
Glaxo vs PAINS	QED	3.518875	0.000476	***
Glaxo vs PAINS	SAscore	0.010826	0.991367	ns
Glaxo vs SureChEMBL	MPO	-3.0933	0.002122	**
Glaxo vs SureChEMBL	QED	-2.49077	0.013092	*
Glaxo vs SureChEMBL	SAscore	-9.82303	1.07E-20	****
Inpharmatica vs LINT	MPO	-9.23105	1.99E-18	****
Inpharmatica vs LINT	QED	-3.72107	0.000235	***
Inpharmatica vs LINT	SAscore	4.291646	2.29E-05	****
Inpharmatica vs MLSMR	MPO	-1.21886	0.223882	ns
Inpharmatica vs MLSMR	QED	-1.33055	0.18425	ns
Inpharmatica vs MLSMR	SAscore	-1.87854	0.061178	ns
Inpharmatica vs PAINS	MPO	-2.95143	0.003343	**

Inpharmatica vs PAINS	QED	2.7268	0.006666	**
Inpharmatica vs PAINS	SAscore	2.719151	0.006837	**
Inpharmatica vs SureChEMBL	MPO	-12.3636	3.87E-28	****
Inpharmatica vs SureChEMBL	QED	-4.85951	1.67E-06	****
Inpharmatica vs SureChEMBL	SAscore	-7.59647	2.14E-13	****
LINT vs MLSMR	MPO	5.280764	2.50E-07	****
LINT vs MLSMR	QED	1.502391	0.134132	ns
LINT vs MLSMR	SAscore	-5.5318	6.68E-08	****
LINT vs PAINS	MPO	5.792976	1.29E-08	****
LINT vs PAINS	QED	6.405601	4.26E-10	****
LINT vs PAINS	SAscore	-1.50295	0.133542	ns
LINT vs SureChEMBL	MPO	-9.39944	2.44E-18	****
LINT vs SureChEMBL	QED	-2.18558	0.029443	*
LINT vs SureChEMBL	SAscore	-11.4014	3.52E-26	****
MLSMR vs PAINS	MPO	-0.95357	0.341015	ns
MLSMR vs PAINS	QED	3.590931	0.000371	***
MLSMR vs PAINS	SAscore	4.202664	3.36E-05	****
MLSMR vs SureChEMBL	MPO	-11.2285	6.83E-25	****
MLSMR vs SureChEMBL	QED	-2.94288	0.003442	**
MLSMR vs SureChEMBL	SAscore	-5.14867	4.02E-07	****
PAINS vs SureChEMBL	MPO	-11.2916	1.43E-24	****
PAINS vs SureChEMBL	QED	-7.03431	6.78E-12	****
PAINS vs SureChEMBL	SAscore	-9.95906	3.99E-21	****

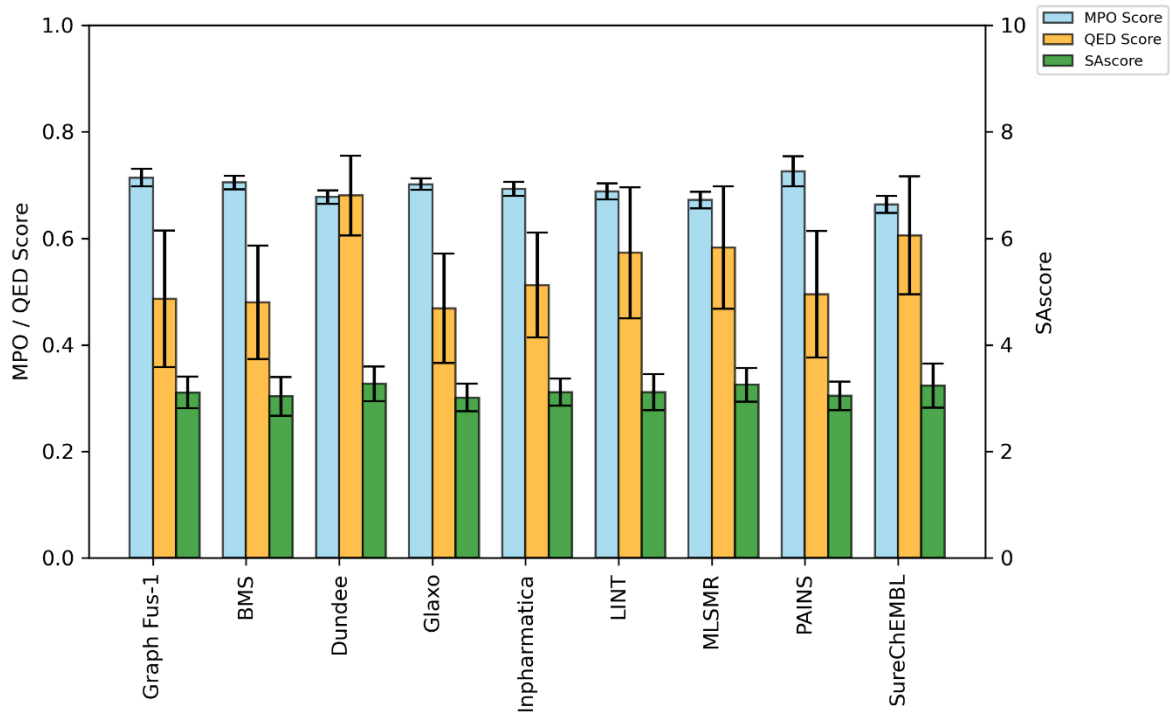


Figure 0-16. The Average Zaleplon MPO Task, SAscore, and QED Scores of the Molecules Generated by the RNN-LSTM Augmented using Graph-fusion 1 with and without Filtering. The results shown are averaged across three repeats. The left y-axis represents the scale of values for the MPO and QED scores. The right y-axis represents the scale of values for the SAscores.

Table 0-24. Welch's t-test Results for the Pairwise Comparison of the RNN-LSTM Augmented using Graph-fusion 1 with and without Filtering when Tasked on the Zaleplon MPO. The significance indicates: **** (p <= 0.0001), *** (p <= 0.001), ** (p <= 0.01), * (p <= 0.05), and ns (p > 0.05).

Comparison	Score	t-statistic	p-value	Significance
Graph Fus-1 vs BMS	MPO	7.492009	2.91E-13	****
Graph Fus-1 vs BMS	QED	0.628249	0.530107	ns
Graph Fus-1 vs BMS	SAscore	2.566333	0.010553	*
Graph Fus-1 vs Dundee	MPO	29.14464	4.35E-111	****
Graph Fus-1 vs Dundee	QED	-21.4897	1.69E-71	****
Graph Fus-1 vs Dundee	SAscore	-6.00838	3.61E-09	****
Graph Fus-1 vs Glaxo	MPO	10.58025	1.28E-23	****
Graph Fus-1 vs Glaxo	QED	1.810731	0.070742	ns
Graph Fus-1 vs Glaxo	SAscore	4.117556	4.41E-05	****
Graph Fus-1 vs Inpharmatica	MPO	17.07513	2.00E-52	****
Graph Fus-1 vs Inpharmatica	QED	-2.69806	0.007199	**
Graph Fus-1 vs Inpharmatica	SAscore	-0.12427	0.901145	ns
Graph Fus-1 vs LINT	MPO	19.8999	4.67E-67	****
Graph Fus-1 vs LINT	QED	-8.22722	1.33E-15	****
Graph Fus-1 vs LINT	SAscore	-0.0309	0.975357	ns
Graph Fus-1 vs MLSMR	MPO	31.70835	3.16E-127	****
Graph Fus-1 vs MLSMR	QED	-9.40129	1.39E-19	****
Graph Fus-1 vs MLSMR	SAscore	-5.68707	2.08E-08	****
Graph Fus-1 vs PAINS	MPO	-6.2476	9.43E-10	****
Graph Fus-1 vs PAINS	QED	-0.86672	0.386465	ns
Graph Fus-1 vs PAINS	SAscore	2.719938	0.006733	**
Graph Fus-1 vs SureChEMBL	MPO	36.86106	1.87E-152	****
Graph Fus-1 vs SureChEMBL	QED	-11.8565	4.86E-29	****
Graph Fus-1 vs SureChEMBL	SAscore	-4.22534	2.82E-05	****
BMS vs Dundee	MPO	25.08364	2.76E-91	****
BMS vs Dundee	QED	-25.0148	8.57E-90	****
BMS vs Dundee	SAscore	-7.8046	3.30E-14	****
BMS vs Glaxo	MPO	3.044384	0.002446	**
BMS vs Glaxo	QED	1.29138	0.197106	ns
BMS vs Glaxo	SAscore	0.877931	0.38041	ns
BMS vs Inpharmatica	MPO	11.10419	4.85E-26	****
BMS vs Inpharmatica	QED	-3.72249	0.000217	***
BMS vs Inpharmatica	SAscore	-2.82393	0.004937	**
BMS vs LINT	MPO	14.61598	3.87E-41	****
BMS vs LINT	QED	-9.60931	2.48E-20	****
BMS vs LINT	SAscore	-2.46016	0.014191	*
BMS vs MLSMR	MPO	28.0171	1.42E-107	****
BMS vs MLSMR	QED	-10.9643	1.82E-25	****
BMS vs MLSMR	SAscore	-7.57367	1.58E-13	****
BMS vs PAINS	MPO	-11.7001	2.10E-27	****
BMS vs PAINS	QED	-1.60731	0.108552	ns

BMS vs PAINS	SAscore	-0.29536	0.767842	ns
BMS vs SureChEMBL	MPO	33.81306	3.71E-135	****
BMS vs SureChEMBL	QED	-13.7117	4.21E-37	****
BMS vs SureChEMBL	SAscore	-6.07443	2.31E-09	****
Dundee vs Glaxo	MPO	-24.4427	2.29E-86	****
Dundee vs Glaxo	QED	27.44341	2.19E-103	****
Dundee vs Glaxo	SAscore	10.08175	9.52E-22	****
Dundee vs Inpharmatica	MPO	-14.2389	3.78E-39	****
Dundee vs Inpharmatica	QED	22.39938	1.19E-78	****
Dundee vs Inpharmatica	SAscore	6.27081	8.26E-10	****
Dundee vs LINT	MPO	-8.83459	1.45E-17	****
Dundee vs LINT	QED	12.38464	8.25E-31	****
Dundee vs LINT	SAscore	5.647783	2.66E-08	****
Dundee vs MLSMR	MPO	4.693445	3.43E-06	****
Dundee vs MLSMR	QED	11.74517	2.83E-28	****
Dundee vs MLSMR	SAscore	0.641666	0.521376	ns
Dundee vs PAINS	MPO	-26.5742	3.81E-91	****
Dundee vs PAINS	QED	21.81443	2.55E-74	****
Dundee vs PAINS	SAscore	8.740945	4.02E-17	****
Dundee vs SureChEMBL	MPO	10.94094	3.03E-25	****
Dundee vs SureChEMBL	QED	9.20871	8.76E-19	****
Dundee vs SureChEMBL	SAscore	1.139822	0.254877	ns
Glaxo vs Inpharmatica	MPO	9.147025	1.11E-18	****
Glaxo vs Inpharmatica	QED	-5.20429	2.71E-07	****
Glaxo vs Inpharmatica	SAscore	-4.61991	4.74E-06	****
Glaxo vs LINT	MPO	13.04153	8.84E-34	****
Glaxo vs LINT	QED	-11.0639	7.25E-26	****
Glaxo vs LINT	SAscore	-3.84583	0.000135	***
Glaxo vs MLSMR	MPO	27.48931	4.85E-102	****
Glaxo vs MLSMR	QED	-12.5162	6.74E-32	****
Glaxo vs MLSMR	SAscore	-10.0553	6.02E-22	****
Glaxo vs PAINS	MPO	-13.7374	6.76E-35	****
Glaxo vs PAINS	QED	-2.88099	0.004115	**
Glaxo vs PAINS	SAscore	-1.43744	0.151136	ns
Glaxo vs SureChEMBL	MPO	33.66793	1.69E-129	****
Glaxo vs SureChEMBL	QED	-15.3831	6.26E-45	****
Glaxo vs SureChEMBL	SAscore	-7.75553	5.36E-14	****
Inpharmatica vs LINT	MPO	4.402021	1.28E-05	****
Inpharmatica vs LINT	QED	-6.56153	1.23E-10	****
Inpharmatica vs LINT	SAscore	0.082099	0.934598	ns
Inpharmatica vs MLSMR	MPO	17.97491	3.02E-57	****
Inpharmatica vs MLSMR	QED	-7.88462	1.67E-14	****
Inpharmatica vs MLSMR	SAscore	-5.97242	4.21E-09	****
Inpharmatica vs PAINS	MPO	-18.22	1.63E-54	****

Inpharmatica vs PAINS	QED	1.860137	0.063395	ns
Inpharmatica vs PAINS	SAscore	3.085741	0.002128	**
Inpharmatica vs SureChEMBL	MPO	24.01473	1.59E-87	****
Inpharmatica vs SureChEMBL	QED	-10.7097	1.70E-24	****
Inpharmatica vs SureChEMBL	SAscore	-4.33729	1.76E-05	****
LINT vs MLSMR	MPO	12.73768	6.78E-33	****
LINT vs MLSMR	QED	-0.97343	0.330754	ns
LINT vs MLSMR	SAscore	-5.30557	1.61E-07	****
LINT vs PAINS	MPO	-20.4016	1.50E-65	****
LINT vs PAINS	QED	7.713658	5.43E-14	****
LINT vs PAINS	SAscore	2.557119	0.010824	*
LINT vs SureChEMBL	MPO	18.53142	2.26E-60	****
LINT vs SureChEMBL	QED	-3.35917	0.000834	***
LINT vs SureChEMBL	SAscore	-4.00496	7.05E-05	****
MLSMR vs PAINS	MPO	-28.8231	5.80E-104	****
MLSMR vs PAINS	QED	8.94093	5.32E-18	****
MLSMR vs PAINS	SAscore	8.611607	7.46E-17	****
MLSMR vs SureChEMBL	MPO	5.980923	3.92E-09	****
MLSMR vs SureChEMBL	QED	-2.44573	0.014757	*
MLSMR vs SureChEMBL	SAscore	0.606681	0.54432	ns
PAINS vs SureChEMBL	MPO	32.6662	1.06E-121	****
PAINS vs SureChEMBL	QED	-11.5207	9.35E-28	****
PAINS vs SureChEMBL	SAscore	-6.59011	1.14E-10	****

Appendix C Case Study

Table 0-1. Welch's t-test Results for the Pairwise Comparison of the baseline RNN-LSTM and the augmented RNN-LSTM (Atom-fusion 3, Atom-fusion 3 with FG mask, Graph—fusion 1) when tasked on the 5-HT_{2A}. The significance indicates: **** (p <= 0.0001), *** (p <= 0.001), ** (p <= 0.01), * (p <= 0.05), and ns (p > 0.05).

Comparison	Score	t-statistic	p-value	Significance
RNN-LSTM vs Atom Fus-3	MPO	3.728834	0.000212	***
RNN-LSTM vs Atom Fus-3	QED	-1.6889	0.091784	ns
RNN-LSTM vs Atom Fus-3	SAscore	4.779383	2.25E-06	****
RNN-LSTM vs Atom Fusion 3 FG mask	MPO	5.948148	4.83E-09	****
RNN-LSTM vs Atom Fusion 3 FG mask	QED	-2.38921	0.017207	*
RNN-LSTM vs Atom Fusion 3 FG mask	SAscore	6.605605	9.05E-11	****
RNN-LSTM vs Graph Fus-1	MPO	-8.29445	8.02E-16	****
RNN-LSTM vs Graph Fus-1	QED	1.488967	0.137058	ns
RNN-LSTM vs Graph Fus-1	SAscore	-7.61486	2.15E-13	****
Atom Fus-3 vs Atom Fusion 3 FG mask	MPO	2.402205	0.016617	*
Atom Fus-3 vs Atom Fusion 3 FG mask	QED	-0.74351	0.457475	ns
Atom Fus-3 vs Atom Fusion 3 FG mask	SAscore	2.175774	0.029981	*
Atom Fus-3 vs Graph Fus-1	MPO	-11.6938	2.03E-28	****
Atom Fus-3 vs Graph Fus-1	QED	3.133055	0.001821	**
Atom Fus-3 vs Graph Fus-1	SAscore	-10.2481	9.10E-22	****
Atom Fusion 3 FG mask vs Graph Fus-1	MPO	-13.337	2.98E-35	****
Atom Fusion 3 FG mask vs Graph Fus-1	QED	3.783546	0.000171	***
Atom Fusion 3 FG mask vs Graph Fus-1	SAscore	-11.2195	2.33E-25	****

Table 0-2. Welch's t-test Results for the Pairwise Comparison of the baseline RNN-LSTM and the augmented RNN-LSTM (Atom-fusion 3, Atom-fusion 3 with FG mask, Graph—fusion 1) when tasked on the 5-HT_{2A}-Serotonin-Dopamine. The significance indicates: **** (p <= 0.0001), *** (p <= 0.001), ** (p <= 0.01), * (p <= 0.05), and ns (p > 0.05).

Comparison	Score	t-statistic	p-value	Significance
RNN-LSTM vs Atom Fus-3	MPO	-5.00088	7.56E-07	****
RNN-LSTM vs Atom Fus-3	QED	-2.403	0.016563	*
RNN-LSTM vs Atom Fus-3	SAscore	-15.6184	7.91E-44	****
RNN-LSTM vs Atom Fusion 1 FG mask	MPO	-0.22947	0.818598	ns
RNN-LSTM vs Atom Fusion 1 FG mask	QED	-5.92667	5.22E-09	****
RNN-LSTM vs Atom Fusion 1 FG mask	SAscore	-16.6503	3.20E-48	****
RNN-LSTM vs Graph Fus-1	MPO	-47.8271	3.45E-192	****
RNN-LSTM vs Graph Fus-1	QED	3.669654	0.000265	***
RNN-LSTM vs Graph Fus-1	SAscore	-22.6362	3.33E-75	****
Atom Fus-3 vs Atom Fusion 1 FG mask	MPO	3.522349	0.000464	***
Atom Fus-3 vs Atom Fusion 1 FG mask	QED	-3.31657	0.000967	***
Atom Fus-3 vs Atom Fusion 1 FG mask	SAscore	-0.96915	0.332858	ns
Atom Fus-3 vs Graph Fus-1	MPO	-40.9888	4.39E-172	****
Atom Fus-3 vs Graph Fus-1	QED	5.890496	6.41E-09	****
Atom Fus-3 vs Graph Fus-1	SAscore	-5.40714	9.24E-08	****
Atom Fusion 1 FG mask vs Graph Fus-1	MPO	-37.9693	2.87E-161	****
Atom Fusion 1 FG mask vs Graph Fus-1	QED	9.397606	1.22E-19	****
Atom Fusion 1 FG mask vs Graph Fus-1	SAscore	-4.39167	1.33E-05	****

Bibliography

Adilov, S. (2021) *GENERATIVE PRE-TRAINING FROM MOLECULES A PREPRINT*. Available at: http://github.com/sanjaradylov/smiles-gpt.

Amabilino, S. *et al.* (2020) 'Guidelines for Recurrent Neural Network Transfer Learning-Based Molecular Generation of Focused Libraries', *Journal of Chemical Information and Modeling*, 60(12), pp. 5699–5713. Available at: https://doi.org/10.1021/acs.jcim.0c00343.

Arús-Pous, J., Blaschke, T., *et al.* (2019) 'Exploring the GDB-13 chemical space using deep generative models', *Journal of Cheminformatics*, 11(1), pp. 1–14. Available at: https://doi.org/10.1186/s13321-019-0341-z.

Arús-Pous, J., Johansson, S.V., *et al.* (2019) 'Randomized SMILES strings improve the quality of molecular generative models', *Journal of Cheminformatics*, 11(1), pp. 1–13. Available at: https://doi.org/10.1186/s13321-019-0393-0.

Ashraf, S.N. *et al.* (2024) 'Hit me with your best shot: Integrated hit discovery for the next generation of drug targets', *Drug Discovery Today*, 29(10), p. 104143. Available at: https://doi.org/https://doi.org/10.1016/j.drudis.2024.104143.

Baell, J.B. and Holloway, G.A. (2010) 'New substructure filters for removal of pan assay interference compounds (PAINS) from screening libraries and for their exclusion in bioassays', *Journal of Medicinal Chemistry*, 53(7), pp. 2719–2740. Available at: https://doi.org/10.1021/jm901137j.

Ban, T.A. (2006) 'The role of serendipity in drug discovery', *Dialogues in Clinical Neuroscience*, 8(3), pp. 335–344. Available at: https://doi.org/10.31887/DCNS.2006.8.3/tban.

Bassani, D. and Moro, S. (2023) 'Past, Present, and Future Perspectives on Computer-Aided Drug Design Methodologies', *Molecules*. Multidisciplinary Digital Publishing Institute (MDPI). Available at: https://doi.org/10.3390/molecules28093906.

Belinkov, Y. and Bisk, Y. (2017) 'Synthetic and Natural Noise Both Break Neural Machine Translation'. Available at: http://arxiv.org/abs/1711.02173.

Bemis, G.W. and Murcko, M.A. (1996) 'The Properties of Known Drugs. 1. Molecular Frameworks', *Journal of Medicinal Chemistry*, 39(15), pp. 2887–2893. Available at: https://doi.org/10.1021/jm9602928.

Bickerton, G.R. *et al.* (2012) 'Quantifying the chemical beauty of drugs', *Nature Chemistry*, 4(2), pp. 90–98. Available at: https://doi.org/10.1038/nchem.1243.

Bjerrum, E.J. (2017) 'SMILES Enumeration as Data Augmentation for Neural Network Modeling of Molecules', (Figure 1). Available at: http://arxiv.org/abs/1703.07076.

Bjerrum, E.J. *et al.* (2023) 'Faster and more diverse de novo molecular optimization with double-loop reinforcement learning using augmented SMILES', *Journal of Computer-Aided Molecular Design* [Preprint]. Available at: https://doi.org/10.1007/s10822-023-00512-6.

Blake, J. (2005) 'Identification and Evaluation of Molecular Properties Related to Preclinical Optimization and Clinical Fate', *Medicinal Chemistry*, 1(6), pp. 649–655. Available at: https://doi.org/10.2174/157340605774598081.

Blanchard, A.E., Stanley, C. and Bhowmik, D. (2021) 'Using GANs with adaptive training data to search for new molecules', *Journal of Cheminformatics*, 13(1), pp. 4–11. Available at: https://doi.org/10.1186/s13321-021-00494-3.

Blaschke, T. *et al.* (2020) 'REINVENT 2.0: An AI Tool for De Novo Drug Design', *Journal of Chemical Information and Modeling* [Preprint]. Available at: https://doi.org/10.1021/acs.jcim.0c00915.

Bleicher, K.H. *et al.* (2003) 'Hit and lead generation: beyond high-throughput screening', *Nature Reviews Drug Discovery*, 2(5), pp. 369–378. Available at: https://doi.org/10.1038/nrd1086.

Bohacek, R.S., McMartin, C. and Guida, W.C. (1996) 'The art and practice of structure-based drug design: A molecular modeling perspective', *Medicinal Research Reviews*, pp. 3–50. Available at: https://doi.org/10.1002/(SICI)1098-1128(199601)16:1<3::AID-MED1>3.0.CO;2-6.

Böhm, H.-J. (1992) 'The computer program LUDI: A new method for the de novo design of enzyme inhibitors', *Journal of Computer-Aided Molecular Design*, 6(1), pp. 61–78. Available at: https://doi.org/10.1007/BF00124387.

Brenk, R. *et al.* (2008) 'Lessons Learnt from Assembling Screening Libraries for Drug Discovery for Neglected Diseases', *ChemMedChem*, 3(3), pp. 435–444. Available at: https://doi.org/10.1002/cmdc.200700139.

Brinkmann, H. *et al.* (2025) 'Going beyond SMILES enumeration for data augmentation in generative drug discovery', *Digital Discovery*, 4(10), pp. 2752–2764. Available at: https://doi.org/10.1039/D5DD00028A.

Brown, N. *et al.* (2004) 'A Graph-Based Genetic Algorithm and Its Application to the Multiobjective Evolution of Median Molecules', *Journal of Chemical Information and Computer Sciences*, 44(3), pp. 1079–1087. Available at: https://doi.org/10.1021/ci034290p.

Brown, N. *et al.* (2019) 'GuacaMol: Benchmarking Models for de Novo Molecular Design', *Journal of Chemical Information and Modeling*, 59(3), pp. 1096–1108. Available at: https://doi.org/10.1021/acs.jcim.8b00839.

CHEMBL24 (2018). Available at: https://doi.org/10.6019/CHEMBL.database.24.

Chen, H. *et al.* (2018) 'The rise of deep learning in drug discovery', *Drug Discovery Today*, 23(6), pp. 1241–1250. Available at: https://doi.org/10.1016/j.drudis.2018.01.039.

Chen, J. *et al.* (2025) 'Data Scaling and Generalization Insights for Medicinal Chemistry Deep Learning Models', *Journal of Chemical Information and Modeling*, 65(12), pp. 5887–5898. Available at: https://doi.org/10.1021/acs.jcim.5c00538.

Cherkasov, A. *et al.* (2014) 'QSAR Modeling: Where Have You Been? Where Are You Going To?', *Journal of Medicinal Chemistry*, 57(12), pp. 4977–5010. Available at: https://doi.org/10.1021/jm4004285.

Cortes-Ciriano, I. and Bender, A. (2015) 'Improved Chemical Structure-Activity Modeling Through Data Augmentation', *Journal of Chemical Information and Modeling*, 55(12), pp. 2682–2692. Available at: https://doi.org/10.1021/acs.jcim.5b00570.

Coulombe, C. (2018) 'Text Data Augmentation Made Simple By Leveraging NLP Cloud APIs'. Available at: http://arxiv.org/abs/1812.04718.

Cubuk, E.D. *et al.* (2019) 'RandAugment: Practical automated data augmentation with a reduced search space'. Available at: http://arxiv.org/abs/1909.13719.

Daval-Frerot, G. and Paris, F. (2020) *WMD at SemEval-2020 Tasks 7 and 11: Assessing humor and propaganda using Unsupervised Data Augmentation*. Online.

David, L. *et al.* (2020) 'Molecular representations in AI-driven drug discovery: a review and practical guide', *Journal of Cheminformatics*, 12(1), pp. 1–22. Available at: https://doi.org/10.1186/s13321-020-00460-5.

Daylight Chemical Information Systems, Inc. (no date) *SMARTS—A Language for Describing Molecular Patterns and SMILES Extensions for Reactions*, *Daylight Theory Manual*. Available at: https://www.daylight.com/dayhtml/doc/theory/theory.smarts.html (Accessed: 26 October 2025).

Degen, J. *et al.* (2008) 'On the art of compiling and using "drug-like" chemical fragment spaces', *ChemMedChem*, 3(10), pp. 1503–1507. Available at: https://doi.org/10.1002/cmdc.200800178.

Delacre, M., Lakens, D. and Leys, C. (2017) 'Why Psychologists Should by Default Use Welch's t-test Instead of Student's t-test (in press for the International Review of Social Psychology).' Available at: https://doi.org/10.31219/osf.io/sbp6k.

Derringer, G. and Suich, R. (1980) 'Simultaneous Optimization of Several Response Variables', *Journal of Quality Technology*, 12(4), pp. 214–219. Available at: https://doi.org/10.1080/00224065.1980.11980968.

DiMasi, J.A., Grabowski, H.G. and Hansen, R.W. (2016) 'Innovation in the pharmaceutical industry: New estimates of R&D costs', *Journal of Health Economics*, 47, pp. 20–33. Available at: https://doi.org/https://doi.org/10.1016/j.jhealeco.2016.01.012.

Dodds, M. *et al.* (2024) 'Sample efficient reinforcement learning with active learning for molecular design', *Chemical Science*, 15(11), pp. 4146–4160. Available at: https://doi.org/10.1039/d3sc04653b.

Du, Y. *et al.* (2024) 'Machine learning-aided generative molecular design', *Nature Machine Intelligence*. Nature Research, pp. 589–604. Available at: https://doi.org/10.1038/s42256-024-00843-5.

Durant, J.L. *et al.* (2002) 'Reoptimization of MDL keys for use in drug discovery', *Journal of Chemical Information and Computer Sciences*, 42(6), pp. 1273–1280. Available at: https://doi.org/10.1021/ci010132r.

Ertl, P., Altmann, E. and McKenna, J.M. (2020) 'The Most Common Functional Groups in Bioactive Molecules and How Their Popularity Has Evolved over Time', *Journal of Medicinal Chemistry*, 63(15), pp. 8408–8418. Available at: https://doi.org/10.1021/acs.jmedchem.0c00754.

Ertl, P. and Schuffenhauer, A. (2009) 'Estimation of synthetic accessibility score of drug-like molecules based on molecular complexity and fragment contributions', *Journal of Cheminformatics*, 1(1), pp. 1–11. Available at: https://doi.org/10.1186/1758-2946-1-8.

Fabbri, A.R. *et al.* (2020) 'Improving Zero and Few-Shot Abstractive Summarization with Intermediate Fine-tuning and Data Augmentation'. Available at: http://arxiv.org/abs/2010.12836.

Feng, S.Y. *et al.* (2020) 'GenAug: Data Augmentation for Finetuning Text Generators', pp. 29–42. Available at: https://doi.org/10.18653/v1/2020.deelio-1.4.

Fink, T. and Raymond, J.L. (2007) 'Virtual exploration of the chemical universe up to 11 atoms of C, N, O, F: Assembly of 26.4 million structures (110.9 million stereoisomers) and analysis for new ring systems, stereochemistry, physicochemical properties, compound classes, and drug discovery', *Journal of Chemical Information and Modeling*, 47(2), pp. 342–353. Available at: https://doi.org/10.1021/ci600423u.

Gao, W. *et al.* (2022) 'Sample Efficiency Matters: A Benchmark for Practical Molecular Optimization'. Available at: http://arxiv.org/abs/2206.12411.

Gao, W. and Coley, C.W. (2020) 'The Synthesizability of Molecules Proposed by Generative Models', *Journal of Chemical Information and Modeling*, 60(12), pp. 5714–5723. Available at: https://doi.org/10.1021/acs.jcim.0c00174.

Gaulton, A. *et al.* (2012) 'ChEMBL: A large-scale bioactivity database for drug discovery', *Nucleic Acids Research*, 40(D1). Available at: https://doi.org/10.1093/nar/gkr777.

Ghiandoni, G.M. *et al.* (2024) 'Augmenting DMTA using predictive AI modelling at AstraZeneca', *Drug Discovery Today*, 29(4), p. 103945. Available at: https://doi.org/https://doi.org/10.1016/j.drudis.2024.103945.

Gillet, V.J. *et al.* (1994) 'SPROUT: Recent Developments in the de Novo Design of Molecules', *Journal of Chemical Information and Computer Sciences*, 34(1), pp. 207–217. Available at: https://doi.org/10.1021/ci00017a027.

Gillet, V.J., Bodkin, M.J. and Hristozov, D. (2013) 'Multiobjective De Novo Design of Synthetically Accessible Compounds', *De novo Molecular Design*. Wiley, pp. 267–285. Available at: https://doi.org/10.1002/9783527677016.ch11.

Gini, G. *et al.* (2019) 'Could deep learning in neural networks improve the QSAR models?', *SAR and QSAR in Environmental Research*, 30(9), pp. 617–642. Available at: https://doi.org/10.1080/1062936X.2019.1650827.

Glen, R.C. and Payne, A.W.R. (1995) 'A genetic algorithm for the automated generation of molecules within constraints', *Journal of Computer-Aided Molecular Design*, 9(2), pp. 181–202. Available at: https://doi.org/10.1007/BF00124408.

Gómez-Bombarelli, R. *et al.* (2018) 'Automatic Chemical Design Using a Data-Driven Continuous Representation of Molecules', *ACS Central Science*, 4(2), pp. 268–276. Available at: https://doi.org/10.1021/acscentsci.7b00572.

Grant, L.L. and Sit, C.S. (2021) '*De novo* molecular drug design benchmarking', *RSC Medicinal Chemistry*, 12(8), pp. 1273–1280. Available at: https://doi.org/10.1039/D1MD00074H.

Grisoni, F. (2023) 'Chemical language models for de novo drug design: Challenges and opportunities', *Current Opinion in Structural Biology*. Elsevier Ltd. Available at: https://doi.org/10.1016/j.sbi.2023.102527.

Grisoni, F. and Schneider, G. (2022) 'De Novo Molecular Design with Chemical Language Models', in A. Heifetz (ed.) *Artificial Intelligence in Drug Design*. New York, NY: Springer US, pp. 207–232. Available at: https://doi.org/10.1007/978-1-0716-1787-8_9.

Guo, J. and Schwaller, P. (2023) 'Beam Enumeration: Probabilistic Explainability For Sample Efficient Self-conditioned Molecular Design'. Available at: http://arxiv.org/abs/2309.13957.

Guo, J. and Schwaller, P. (2024) 'Augmented Memory: Sample-Efficient Generative Molecular Design with Reinforcement Learning', *JACS Au* [Preprint]. Available at: https://doi.org/10.1021/jacsau.4c00066.

Gupta, A. *et al.* (2018) 'Generative Recurrent Networks for De Novo Drug Design', *Molecular Informatics*, 37(1). Available at: https://doi.org/10.1002/minf.201700111.

Hann, M. *et al.* (1999) 'Strategic Pooling of Compounds for High-Throughput Screening', *Journal of Chemical Information and Computer Sciences*, 39(5), pp. 897–902. Available at: https://doi.org/10.1021/ci990423o.

Hansch, C. *et al.* (1962) 'Correlation of Biological Activity of Phenoxyacetic Acids with Hammett Substituent Constants and Partition Coefficients', *Nature*, 194(4824), pp. 178–180. Available at: https://doi.org/10.1038/194178b0.

Hartenfeller, M. and Schneider, G. (2011) 'Enabling future drug discovery by de novo design', *Wiley Interdisciplinary Reviews: Computational Molecular Science*, 1(5), pp. 742–759. Available at: https://doi.org/10.1002/wcms.49.

Hochreiter, S. and Schmidhuber, J. (1997) 'Long Short-Term Memory', *Neural Computation*, 9(8), pp. 1735–1780. Available at: https://doi.org/10.1162/neco.1997.9.8.1735.

Hodos, R.A. *et al.* (2016) 'In silico methods for drug repurposing and pharmacology', *Wiley Interdisciplinary Reviews: Systems Biology and Medicine*, 8(3), pp. 186–210. Available at: https://doi.org/10.1002/wsbm.1337.

Hughes, J.P. *et al.* (2011) 'Principles of early drug discovery', *British Journal of Pharmacology*, 162(6), pp. 1239–1249. Available at: https://doi.org/10.1111/j.1476-5381.2010.01127.x.

Irwin, J.J. and Shoichet, B.K. (2005) 'ZINC-A Free Database of Commercially Available Compounds for Virtual Screening'. Available at: https://doi.org/10.1021/ci049714.

Ishida, S., Sheppard, T. and Nishikata, T. (2018) 'Site selectivities in fluorination', *Tetrahedron Letters*, 59(9), pp. 789–798. Available at: https://doi.org/10.1016/j.tetlet.2018.01.044.

Jaeger, S., Fulle, S. and Turk, S. (2018) 'Mol2vec: Unsupervised Machine Learning Approach with Chemical Intuition', *Journal of Chemical Information and Modeling*, 58(1), pp. 27–35. Available at: https://doi.org/10.1021/acs.jcim.7b00616.

Jensen, J.H. (2019) 'A graph-based genetic algorithm and generative model/Monte Carlo tree search for the exploration of chemical space', *Chemical Science*, 10(12), pp. 3567–3572. Available at: https://doi.org/10.1039/c8sc05372c.

Jiménez-Luna, J. *et al.* (2021) 'Artificial intelligence in drug discovery: recent advances and future perspectives', *Expert Opinion on Drug Discovery*, 00(00), pp. 1–11. Available at: https://doi.org/10.1080/17460441.2021.1909567.

Johnson, M.A. and Maggiora, G.M. (1990) *Concepts and Applications of Molecular Similarity*, *Journal of Molecular Structure*. Available at: https://doi.org/10.1016/0022-2860(92)85011-5.

Jorgensen, W.L. (2009) 'Efficient Drug Lead Discovery and Optimization', *Accounts of Chemical Research*, 42(6), pp. 724–733. Available at: https://doi.org/10.1021/ar800236t.

Joshi, S. and Srivastava, R. (2023) 'Effect of "magic chlorine" in drug discovery: an *in silico* approach', *RSC Advances*, 13(49), pp. 34922–34934. Available at: https://doi.org/10.1039/D3RA06638J.

Katsila, T. *et al.* (2016) 'Computational approaches in target identification and drug discovery', *Computational and Structural Biotechnology Journal*. Elsevier B.V., pp. 177–184. Available at: https://doi.org/10.1016/j.csbj.2016.04.004.

Keserű, G.M. and Makara, G.M. (2006) 'Hit discovery and hit-to-lead approaches', *Drug Discovery Today*, 11(15), pp. 741–748. Available at: https://doi.org/https://doi.org/10.1016/j.drudis.2006.06.016.

Kim, S. (2009) 'Synthesis and Structural Analysis of One-Dimensional sp-Hybridized Carbon Chain Molecules', *Angewandte Chemie International Edition*, 48(42), pp. 7740–7743. Available at: https://doi.org/10.1002/anie.200904145.

Kimber, T.B., Gagnebin, M. and Volkamer, A. (2021) 'Maxsmi: Maximizing molecular property prediction performance with confidence estimation using SMILES augmentation and deep learning', *Artificial Intelligence in the Life Sciences*, 1, p. 100014. Available at: https://doi.org/10.1016/j.ailsci.2021.100014.

Kumar, A., Tiwari, A. and Sharma, A. (2018) 'Changing Paradigm from one Target one Ligand Towards Multi-target Directed Ligand Design for Key Drug Targets of Alzheimer Disease: An Important Role of In Silico Methods in Multi-target Directed Ligands Design', *Current Neuropharmacology*, 16(6), pp. 726–739. Available at: https://doi.org/10.2174/1570159x16666180315141643.

Kuntz, I.D. *et al.* (1999) 'The maximal affinity of ligands', *Proceedings of the National Academy of Sciences*, 96(18), pp. 9997–10002. Available at: https://doi.org/10.1073/pnas.96.18.9997.

Langdon, S.R., Ertl, P. and Brown, N. (2010) 'Bioisosteric Replacement and Scaffold Hopping in Lead Generation and Optimization', *Molecular Informatics*, 29(5), pp. 366–385. Available at: https://doi.org/10.1002/minf.201000019.

Leach, A.R. and Gillet, V.J. (2007) *An Introduction To Chemoinformatics*, *Springer*. Available at: https://doi.org/https://doi.org/10.1007/978-1-4020-6291-9.

Lecun, Y., Bengio, Y. and Hinton, G. (2015) 'Deep learning', *Nature*, 521(7553), pp. 436–444. Available at: https://doi.org/10.1038/nature14539.

Li, B., Hou, Y. and Che, W. (2022) 'Data augmentation approaches in natural language processing: A survey', *AI Open*, 3, pp. 71–90. Available at: https://doi.org/10.1016/j.aiopen.2022.03.001.

Li, X. *et al.* (2020) 'Chemical space exploration based on recurrent neural networks: Applications in discovering kinase inhibitors', *Journal of Cheminformatics*, 12(1), pp. 1–13. Available at: https://doi.org/10.1186/s13321-020-00446-3.

Li, X. and Fourches, D. (2021) 'SMILES Pair Encoding: A Data-Driven Substructure Tokenization Algorithm for Deep Learning', *Journal of Chemical Information and Modeling*, 61(4), pp. 1560–1569. Available at: https://doi.org/10.1021/acs.jcim.0c01127.

Li, Y., Zhang, L. and Liu, Z. (2018) 'Multi-objective de novo drug design with conditional graph generative model', *Journal of Cheminformatics*, 10(1), pp. 1–24. Available at: https://doi.org/10.1186/s13321-018-0287-6.

Lipinski, C. and Hopkins, A. (2004) 'Navigating chemical space for biology and medicine', *Nature*, 432(7019), pp. 855–861. Available at: https://doi.org/10.1038/nature03193.

Liu, X. *et al.* (2019) 'An exploration strategy improves the diversity of de novo ligands using deep reinforcement learning: A case for the adenosine A2A receptor', *Journal of Cheminformatics*, 11(1), pp. 1–16. Available at: https://doi.org/10.1186/s13321-019-0355-6.

Liu, X. *et al.* (2021) 'DrugEx v2: de novo design of drug molecules by Pareto-based multi-objective reinforcement learning in polypharmacology', *Journal of Cheminformatics*, 13(1). Available at: https://doi.org/10.1186/s13321-021-00561-9.

Liu, X., IJzerman, A.P. and van Westen, G.J.P. (2021) 'Computational Approaches for De Novo Drug Design: Past, Present, and Future', *Methods in Molecular Biology*. Humana Press Inc., pp. 139–165. Available at: https://doi.org/10.1007/978-1-0716-0826-5_6.

Lo, S. *et al.* (2023) 'Augmenting Polymer Datasets by Iterative Rearrangement', *Journal of Chemical Information and Modeling*, 63(14), pp. 4266–4276. Available at: https://doi.org/10.1021/acs.jcim.3c00144.

Loeffler, H.H. *et al.* (2024) 'Reinvent 4: Modern AI–driven generative molecule design', *Journal of Cheminformatics*, 16(1), p. 20. Available at: https://doi.org/10.1186/s13321-024-00812-5.

Longpre, S., Wang, Y. and DuBois, C. (2020) 'How Effective is Task-Agnostic Data Augmentation for Pretrained Transformers?', *Findings of the Association for Computational Linguistics: EMNLP 2020*. Stroudsburg, PA, USA: Association for Computational Linguistics, pp. 4401–4411. Available at: https://doi.org/10.18653/v1/2020.findings-emnlp.394.

Lowell, D. *et al.* (2020) 'Unsupervised Data Augmentation with Naive Augmentation and without Unlabeled Data'. Available at: http://arxiv.org/abs/2010.11966.

Macalino, S.J.Y. *et al.* (2015) 'Role of computer-aided drug design in modern drug discovery', *Archives of Pharmacal Research*, 38(9), pp. 1686–1701. Available at: https://doi.org/10.1007/s12272-015-0640-5.

Magar, R. *et al.* (2021) 'AugLiChem: Data Augmentation Library of Chemical Structures for Machine Learning'. Available at: http://arxiv.org/abs/2111.15112.

Maggiora, G. *et al.* (2014) 'Molecular Similarity in Medicinal Chemistry', *Journal of Medicinal Chemistry*, 57(8), pp. 3186–3204. Available at: https://doi.org/10.1021/jm401411z.

Medina-Franco, J.L. *et al.* (2022) 'Progress on open chemoinformatic tools for expanding and exploring the chemical space', *Journal of Computer-Aided Molecular Design*, 36(5), pp. 341–354. Available at: https://doi.org/10.1007/s10822-021-00399-1.

Merk, D. *et al.* (2018) 'De Novo Design of Bioactive Small Molecules by Artificial Intelligence', *Molecular Informatics*, 37(1), pp. 3–6. Available at: https://doi.org/10.1002/minf.201700153.

Mervin, L.H. *et al.* (2015) 'Target prediction utilising negative bioactivity data covering large chemical space', *Journal of Cheminformatics*, 7(1), p. 51. Available at: https://doi.org/10.1186/s13321-015-0098-y.

Meyers, J., Fabian, B. and Brown, N. (2021) 'De novo molecular design and generative models', *Drug Discovery Today*, 26(11), pp. 2707–2715. Available at: https://doi.org/10.1016/j.drudis.2021.05.019.

Moret, M. *et al.* (2020) 'Generative molecular design in low data regimes', *Nature Machine Intelligence*, 2(3), pp. 171–180. Available at: https://doi.org/10.1038/s42256-020-0160-y.

Morgan, H.L. (1965) 'The Generation of a Unique Machine Description for Chemical Structures—A Technique Developed at Chemical Abstracts Service', *Journal of Chemical Documentation*, 5(2), pp. 107–113. Available at: https://doi.org/10.1021/c160017a018.

Müller, S.G. and Hutter, F. (2021) 'TrivialAugment: Tuning-free Yet State-of-the-Art Data Augmentation'. Available at: http://arxiv.org/abs/2103.10158.

Muratov, E.N. *et al.* (2020) 'QSAR without borders', *Chemical Society Reviews*, 49(11), pp. 3525–3564. Available at: https://doi.org/10.1039/D0CS00098A.

Neil, D. *et al.* (2018) 'Exploring deep recurrent models with reinforcement learning for molecule design', *6th International Conference on Learning Representations, ICLR 2018 - Workshop Track Proceedings*, pp. 1–15.

NIH Molecular Libraries (2013) *MLSMR Excluded Functionality Filters*. Available at: https://www.yumpu.com/en/document/read/12367541/mlsmr-excluded-functionality-filters-nih-molecular-libraries- (Accessed: 3 January 2025).

Nittinger, E. (2023) 'Generative Drug Design with REINVENT - Possibilities and Open Challenges'.

Olivecrona, M. *et al.* (2017) 'Molecular de-novo design through deep reinforcement learning', *Journal of Cheminformatics*, 9(1), pp. 1–14. Available at: https://doi.org/10.1186/s13321-017-0235-x.

Patani, G.A. and LaVoie, E.J. (1996) 'Bioisosterism: A Rational Approach in Drug Design', *Chemical Reviews*, 96(8), pp. 3147–3176. Available at: https://doi.org/10.1021/cr950066q.

Pearce, B.C. *et al.* (2006) 'An Empirical Process for the Design of High-Throughput Screening Deck Filters', *Journal of Chemical Information and Modeling*, 46(3), pp. 1060–1068. Available at: https://doi.org/10.1021/ci050504m.

Pearlman, D.A. and Murcko, M.A. (1993) 'CONCEPTS: New dynamic algorithm for de novo drug suggestion', *Journal of Computational Chemistry*, 14(10), pp. 1184–1193. Available at: https://doi.org/10.1002/jcc.540141008.

Peng, B. *et al.* (2020) 'Data Augmentation for Spoken Language Understanding via Pretrained Language Models'. Available at: http://arxiv.org/abs/2004.13952.

Plowright, A.T. *et al.* (2012) 'Hypothesis driven drug design: Improving quality and effectiveness of the design-make-test-analyse cycle', *Drug Discovery Today*, 17(1–2), pp. 56–62. Available at: https://doi.org/10.1016/j.drudis.2011.09.012.

Poduri, R. (2021) 'Historical Perspective of Drug Discovery and Development', *Drug Discovery and Development*. Singapore: Springer Singapore, pp. 1–10. Available at: https://doi.org/10.1007/978-981-15-5534-3_1.

Polykovskiy, D. *et al.* (2020) 'Molecular Sets (MOSES): A Benchmarking Platform for Molecular Generation Models', *Frontiers in Pharmacology*, 11, pp. 1–19. Available at: https://doi.org/10.3389/fphar.2020.565644.

Popova, M. *et al.* (2019) 'MolecularRNN: Generating realistic molecular graphs with optimized properties'.

Popova, M., Isayev, O. and Tropsha, A. (2018) 'Deep reinforcement learning for de novo drug design', *Science Advances*, 4(7), pp. 1–14. Available at: https://doi.org/10.1126/sciadv.aap7885.

Preuer, K. *et al.* (2018) 'Fréchet ChemNet Distance: A Metric for Generative Models for Molecules in Drug Discovery', *Journal of Chemical Information and Modeling*, 58(9), pp. 1736–1741. Available at: https://doi.org/10.1021/acs.jcim.8b00234.

Radford, A. *et al.* (2019) 'Language Models are Unsupervised Multitask Learners'.

Rastogi, C., Mofid, N. and Hsiao, F.-I. (2020) 'Can We Achieve More with Less? Exploring Data Augmentation for Toxic Comment Classification'. Available at: http://arxiv.org/abs/2007.00875.

Regina, M., Meyer, M. and Goutal, S. (2020) 'Text Data Augmentation: Towards better detection of spear-phishing emails'. Available at: http://arxiv.org/abs/2007.02033.

Renz, P. *et al.* (2019) 'On failure modes in molecule generation and optimization', *Drug Discovery Today: Technologies*, 32–33, pp. 55–63. Available at: https://doi.org/10.1016/j.ddtec.2020.09.003.

Reymond, J.L. *et al.* (2012) 'The enumeration of chemical space', *Wiley Interdisciplinary Reviews: Computational Molecular Science*, pp. 717–733. Available at: https://doi.org/10.1002/wcms.1104.

Rogers, D. and Hahn, M. (2010) 'Extended-connectivity fingerprints', *Journal of Chemical Information and Modeling*, 50(5), pp. 742–754. Available at: https://doi.org/10.1021/ci100050t.

Ruddigkeit, L. *et al.* (2012) 'Enumeration of 166 billion organic small molecules in the chemical universe database GDB-17', *Journal of Chemical Information and Modeling*, 52(11), pp. 2864–2875. Available at: https://doi.org/10.1021/ci300415d.

Rumelhart, D.E., Hinton, G.E. and Williams, R.J. (1986) 'Learning representations by back-propagating errors', *Nature*, 323(6088), pp. 533–536. Available at: https://doi.org/10.1038/323533a0.

Schaller, D. *et al.* (2020) 'Next generation 3D pharmacophore modeling', *WIREs Computational Molecular Science*, 10(4). Available at: https://doi.org/10.1002/wcms.1468.

Schenone, M. *et al.* (2013) 'Target identification and mechanism of action in chemical biology and drug discovery', *Nature Chemical Biology*, pp. 232–240. Available at: https://doi.org/10.1038/nchembio.1199.

Schneider, G. *et al.* (2000) 'Virtual Screening for Bioactive Molecules by Evolutionary De Novo Design', *Angewandte Chemie*, 39(22), pp. 4130–4133. Available at: https://doi.org/10.1002/1521-3773(20001117)39:22<4130::AID-ANIE4130>3.0.CO;2-E.

Schneider, G. and Baringhaus, K.-H. (2008) *Molecular Design: Concepts and Applications*. Weinheim: Wiley-VCH.

Schneider, G. and Clark, D.E. (2019) 'Automated De Novo Drug Design: Are We Nearly There Yet?', *Angewandte Chemie - International Edition*, 58(32), pp. 10792–10803. Available at: https://doi.org/10.1002/anie.201814681.

Schneider, G., Hartenfeller, M. and Proschak, E. (2010) 'De novo Drug Design', *Lead Generation Approaches in Drug Discovery*, 33(6–7), pp. 165–185. Available at: https://doi.org/10.1002/9780470584170.ch6.

Schneider, P. *et al.* (2020) 'Rethinking drug design in the artificial intelligence era', *Nature Reviews Drug Discovery*, 19(5), pp. 353–364. Available at: https://doi.org/10.1038/s41573-019-0050-3.

Scott, K.A., Cox, P.B. and Njardarson, J.T. (2022) 'Phenols in Pharmaceuticals: Analysis of a Recurring Motif', *Journal of Medicinal Chemistry*, 65(10), pp. 7044–7072. Available at: https://doi.org/10.1021/acs.jmedchem.2c00223.

Segler, M.H.S. *et al.* (2018) 'Generating focused molecule libraries for drug discovery with recurrent neural networks', *ACS Central Science*, 4(1), pp. 120–131. Available at: https://doi.org/10.1021/acscentsci.7b00512.

Sennrich, R., Haddow, B. and Birch, A. (2016) 'Neural Machine Translation of Rare Words with Subword Units'.

Shorten, C. and Khoshgoftaar, T.M. (2019) 'A survey on Image Data Augmentation for Deep Learning', *Journal of Big Data*, 6(1). Available at: https://doi.org/10.1186/s40537-019-0197-0.

Shorten, C., Khoshgoftaar, T.M. and Furht, B. (2021) *Text Data Augmentation for Deep Learning*, *Journal of Big Data*. Springer International Publishing. Available at: https://doi.org/10.1186/s40537-021-00492-0.

Šícho, M. *et al.* (2023) 'DrugEx: Deep Learning Models and Tools for Exploration of Drug-Like Chemical Space', *Journal of Chemical Information and Modeling* [Preprint]. Available at: https://doi.org/10.1021/acs.jcim.3c00434.

Singh, N. *et al.* (2023) 'Drug discovery and development: introduction to the general public and patient groups', *Frontiers in Drug Discovery*, 3. Available at: https://doi.org/10.3389/fddsv.2023.1201419.

Skinnider, M.A. (2024) 'Invalid SMILES are beneficial rather than detrimental to chemical language models', *Nature Machine Intelligence* [Preprint]. Available at: https://doi.org/10.1038/s42256-024-00821-x.

Sliwoski, G. *et al.* (2014) 'Computational methods in drug discovery', *Pharmacological Reviews*, 66(1), pp. 334–395. Available at: https://doi.org/10.1124/pr.112.007336.

Spiegel, J.O. and Durrant, J.D. (2020) 'AutoGrow4: an open-source genetic algorithm for de novo drug design and lead optimization', *Journal of Cheminformatics*, 12(1), p. 25. Available at: https://doi.org/10.1186/s13321-020-00429-4.

Ståhl, N. *et al.* (2019) 'Deep Reinforcement Learning for Multiparameter Optimization in de novo Drug Design', *Journal of Chemical Information and Modeling*, 59(7), pp. 3166–3176. Available at: https://doi.org/10.1021/acs.jcim.9b00325.

Stanley, M. and Segler, M. (2023) 'Fake it until you make it? Generative de novo design and virtual screening of synthesizable molecules', *Current Opinion in Structural Biology*, 82, p. 102658. Available at: https://doi.org/10.1016/j.sbi.2023.102658.

Tabana, Y. *et al.* (2023) 'Target identification of small molecules: an overview of the current applications in drug discovery', *BMC Biotechnology*. BioMed Central Ltd. Available at: https://doi.org/10.1186/s12896-023-00815-4.

Tan, X. *et al.* (2020) 'Automated design and optimization of multitarget schizophrenia drug candidates by deep learning', *European Journal of Medicinal Chemistry*, 204, p. 112572. Available at: https://doi.org/10.1016/j.ejmech.2020.112572.

Thomas, M. *et al.* (2022) 'Applications of Artificial Intelligence in Drug Design: Opportunities and Challenges', in A. Heifetz (ed.) *Artificial Intelligence in Drug Design*. New York, NY: Springer US, pp. 1–59. Available at: https://doi.org/10.1007/978-1-0716-1787-8_1.

Thomas, M. *et al.* (2024) 'MolScore: a scoring, evaluation and benchmarking framework for generative models in de novo drug design', *Journal of Cheminformatics*, 16(1), p. 64. Available at: https://doi.org/10.1186/s13321-024-00861-w.

Van Tilborg, D. *et al.* (2024) 'Deep learning for low-data drug discovery: hurdles and opportunities', *ChemRxiv* [Preprint]. Available at: https://doi.org/10.26434/chemrxiv-2024-w0wvl.

Tingle, B.I. *et al.* (2023) 'ZINC-22–A Free Multi-Billion-Scale Database of Tangible Compounds for Ligand Discovery', *Journal of Chemical Information and Modeling*, 63(4), pp. 1166–1176. Available at: https://doi.org/10.1021/acs.jcim.2c01253.

Todorov, N.P. and Dean, P.M. (1997) 'Evaluation of a method for controlling molecular scaffold diversity in de novo ligand design', *Journal of Computer-Aided Molecular Design*, 11(2), pp. 175–192. Available at: https://doi.org/10.1023/A:1008042711516.

Tossou, P. *et al.* (2024) 'Real-World Molecular Out-Of-Distribution: Specification and Investigation', *Journal of Chemical Information and Modeling*, 64(3), pp. 697–711. Available at: https://doi.org/10.1021/acs.jcim.3c01774.

Tripp, A. and Hernández-Lobato, J.M. (2023) 'Genetic algorithms are strong baselines for molecule generation'. Available at: http://arxiv.org/abs/2310.09267.

Tropsha, A. *et al.* (2024) 'Integrating QSAR modelling and deep learning in drug discovery: the emergence of deep QSAR', *Nature Reviews Drug Discovery*, 23(2), pp. 141–155. Available at: https://doi.org/10.1038/s41573-023-00832-0.

Ucak, U. V., Ashyrmamatov, I. and Lee, J. (2023) 'Improving the quality of chemical language model outcomes with atom-in-SMILES tokenization', *Journal of Cheminformatics*, 15(1). Available at: https://doi.org/10.1186/s13321-023-00725-9.

Vanhaelen, Q., Lin, Y.C. and Zhavoronkov, A. (2020) 'The Advent of Generative Chemistry', *ACS Medicinal Chemistry Letters*, 11(8), pp. 1496–1505. Available at: https://doi.org/10.1021/acsmedchemlett.0c00088.

Vaswani, A. *et al.* (2017) 'Attention Is All You Need'. Available at: http://arxiv.org/abs/1706.03762.

Verma, S. and Pathak, R.K. (2022) 'Chapter 16 - Discovery and optimization of lead molecules in drug designing', in D.B. Singh and R.K. Pathak (eds.) *Bioinformatics*. Academic Press, pp. 253–267. Available at: https://doi.org/https://doi.org/10.1016/B978-0-323-89775-4.00004-3.

Vincent Rajkumar, S. (2020) 'The high cost of prescription drugs: causes and solutions', *Blood Cancer Journal*. Springer Nature. Available at: https://doi.org/10.1038/s41408-020-0338-x.

Vinkers, H.M. *et al.* (2003) 'SYNOPSIS: SYNthesize and OPtimize System in Silico', *Journal of Medicinal Chemistry*, 46(13), pp. 2765–2773. Available at: https://doi.org/10.1021/jm030809x.

Virshup, A.M. *et al.* (2013) 'Stochastic Voyages into Uncharted Chemical Space Produce a Representative Library of All Possible Drug-Like Compounds', *Journal of the American Chemical Society*, 135(19), pp. 7296–7303. Available at: https://doi.org/10.1021/ja401184g.

Walter, M., Webb, S.J. and Gillet, V.J. (2024) 'Interpreting Neural Network Models for Toxicity Prediction by Extracting Learned Chemical Features', *Journal of Chemical Information and Modeling*, 64(9), pp. 3670–3688. Available at: https://doi.org/10.1021/acs.jcim.4c00127.

Walters, P. (2018) 'rd_filters'. Available at: https://github.com/PatWalters/rd_filters.

Walters, W.P. (2019) 'Virtual Chemical Libraries', *Journal of Medicinal Chemistry*. American Chemical Society, pp. 1116–1124. Available at: https://doi.org/10.1021/acs.jmedchem.8b01048.

Walters, W.P. and Murcko, M. (2020) 'Assessing the impact of generative AI on medicinal chemistry', *Nature Biotechnology*, 38(2), pp. 143–145. Available at: https://doi.org/10.1038/s41587-020-0418-2.

Walters, W.P. and Namchuk, M. (2003) 'Designing screens: how to make your hits a hit', *Nature Reviews Drug Discovery*, 2(4), pp. 259–266. Available at: https://doi.org/10.1038/nrd1063.

Wang, W.Y. and Yang, D. (2015) *That's So Annoying!!!: A Lexical and Frame-Semantic Embedding Based Data Augmentation Approach to Automatic Categorization of Annoying Behaviors using #petpeeve Tweets **. Association for Computational Linguistics. Available at: http://www.cs.cmu.edu/.

Wang, X. *et al.* (2018) 'SwitchOut: an Efficient Data Augmentation Algorithm for Neural Machine Translation'. Available at: http://arxiv.org/abs/1808.07512.

Waring, M.J. *et al.* (2015) 'An analysis of the attrition of drug candidates from four major pharmaceutical companies', *Nature Reviews Drug Discovery*. Nature Publishing Group, pp. 475–486. Available at: https://doi.org/10.1038/nrd4609.

Warr, W.A. *et al.* (2022) 'Exploration of Ultralarge Compound Collections for Drug Discovery', *Journal of Chemical Information and Modeling* [Preprint]. American Chemical Society. Available at: https://doi.org/10.1021/acs.jcim.2c00224.

Webster, J. (2021) *Development of a Novel Tree Search Algorithm for Reaction Vector Based De Novo Design*. University of Sheffield.

Wei, J. and Zou, K. (2019) 'EDA: Easy Data Augmentation Techniques for Boosting Performance on Text Classification Tasks', *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Stroudsburg, PA, USA: Association for Computational Linguistics, pp. 6381–6387. Available at: https://doi.org/10.18653/v1/D19-1670.

Weininger, D. (1988) 'SMILES, a chemical language and information system. 1. Introduction to methodology and encoding rules', *Journal of Chemical Information and Modeling*, 28(1), pp. 31–36. Available at: https://doi.org/10.1021/ci00057a005.

Weininger, D. (1995) 'Method and apparatus for designing molecules with desired properties by evolving successive populations'. UnitedStates.

Wesolowski, S.S. and Brown, D.G. (2016) 'The Strategies and Politics of Successful Design, Make, Test, and Analyze (DMTA) Cycles in Lead Generation', pp. 487–512. Available at: https://doi.org/10.1002/9783527677047.ch17.

Wiswesser, W.J. (1954) *A Line-formula Chemical Notation*. Crowell. Available at: https://books.google.co.uk/books?id=zh0FAAAAMAAJ.

Wyatt, P.G. *et al.* (2011) *Target Validation: Linking Target and Chemical Properties to Desired Product Profile*, *Current Topics in Medicinal Chemistry*. Available at: http://www.mmv.org/.

Xia, X. *et al.* (2019) 'Graph-based generative models for de Novo drug design', *Drug Discovery Today: Technologies*, 32–33, pp. 45–53. Available at: https://doi.org/10.1016/j.ddtec.2020.11.004.

Xie, Q. *et al.* (2020) 'Unsupervised Data Augmentation for Consistency Training', in H. Larochelle et al. (eds.) *Advances in Neural Information Processing Systems*. Curran Associates, Inc., pp. 6256–6268. Available at: https://proceedings.neurips.cc/paper_files/paper/2020/file/44feb0096faa8326192570788b38c1d1-Paper.pdf.

Xie, Z. *et al.* (2017) 'Data Noising as Smoothing in Neural Network Language Models'. Available at: http://arxiv.org/abs/1703.02573.

Xu, Y. *et al.* (2019) 'Deep learning for molecular generation', *Future Medicinal Chemistry*, 11(6), pp. 567–597. Available at: https://doi.org/10.4155/fmc-2018-0358.

Yan, G. *et al.* (2019) 'Data Augmentation for Deep Learning of Judgment Documents', in J. and Z.S. and X.L. and Y.J. Cui Zhen and Pan (ed.) *Intelligence Science and Big Data Engineering. Big Data and Machine Learning*. Cham: Springer International Publishing, pp. 232–242. Available at: https://doi.org/10.1007/978-3-030-36204-1_19.

Yang, X. *et al.* (2019) 'Concepts of Artificial Intelligence for Computer-Assisted Drug Discovery', *Chemical Reviews*, 119(18), pp. 10520–10594. Available at: https://doi.org/10.1021/acs.chemrev.8b00728.

Yang, X. *et al.* (2025) 'Carboxamide: A Privileged Pharmacophore for the Development of Anti-infectious and Anti-cancer Drugs', *Current Topics in Medicinal Chemistry*, 25. Available at: https://doi.org/10.2174/0115680266363457250714113345.

Yang, X. and Specht, C.G. (2019) 'Subsynaptic domains in super-resolution microscopy: The treachery of images', *Frontiers in Molecular Neuroscience*, 12(July). Available at: https://doi.org/10.3389/fnmol.2019.00161.

Yu, A.W. *et al.* (2018) 'QANet: Combining Local Convolution with Global Self-Attention for Reading Comprehension'. Available at: http://arxiv.org/abs/1804.09541.

Yu, S. *et al.* (2019) 'Hierarchical data augmentation and the application in text classification', *IEEE Access*, 7, pp. 185476–185485. Available at: https://doi.org/10.1109/ACCESS.2019.2960263.

Yuan, W. *et al.* (2017) 'Chemical Space Mimicry for Drug Discovery', *Journal of Chemical Information and Modeling*, 57(4), pp. 875–882. Available at: https://doi.org/10.1021/acs.jcim.6b00754.

Zhang, D. *et al.* (2020) 'On Data Augmentation for Extreme Multi-label Classification'. Available at: http://arxiv.org/abs/2009.10778.

Zhang, Y., Ge, T. and Sun, X. (2020) 'Parallel Data Augmentation for Formality Style Transfer', *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Stroudsburg, PA, USA: Association for Computational Linguistics, pp. 3221–3228. Available at: https://doi.org/10.18653/v1/2020.acl-main.294.

Zhou, Y. *et al.* (2024) 'A Survey on Data Augmentation in Large Model Era'.