CHALMERS
UNIVERSITY OF TECHNOLOGY

UNIVERSITY OF GOTHENBURG

Contrastive Learning For Molecular Representation

Learning Molecular Representations via Contrastive Learning: A Deep Learning Approach

Master's thesis in Computer science and engineering

SALAM HANI, JONATHAN LINDER

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2025

Master's thesis 2025

Contrastive Learning For Molecular Representation

Learning Molecular Representations via Contrastive Learning: A Deep Learning Approach

SALAM HANI, JONATHAN LINDER

UNIVERSITY OF GOTHENBURG

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Computer Science and Engineering
Chalmers University of Technology
University of Gothenburg
Gothenburg, Sweden 2025

Contrastive Learning For Molecular Representation LaTeX
Learning Molecular Representations via Contrastive Learning: A Deep Learning Approach
SALAM HANI, JONATHAN LINDER

Supervisor: Hannes Löffler, AstraZeneca IT R&D
Advisor: Simon Olsson, Chalmers, Dep. Computer Science
Examiner: Jean-Phillippe Bernardy, Dep. Computer Science

Master's Thesis 2025
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: Description of the picture on the cover page (if applicable)

Typeset in LaTeX
Gothenburg, Sweden 2025

Contrastive Learning For Molecular Representation LaTeX
Learning Molecular Representations via Contrastive Learning: A Deep Learning Approach
SALAM HANI, JONATHAN LINDER
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg

Abstract

This thesis explores the integration of contrastive learning into REINVENT, AstraZeneca's in-house generative model for molecular design, with the aim of improving the model's understanding of chemical equivalence between different SMILES representations of the same compound. To this end, a contrastive learning framework was developed, incorporating SMILES-based data augmentation techniques such as enumeration and subgraphing.

The framework was evaluated on three datasets: a proprietary baseline derived from ChEMBL35, and the publicly available MOSES and GuacaMol datasets. To assess the impact of architectural design on performance, multiple model architectures were investigated, including a newly introduced intermediate architecture.

Results indicate that the intermediate architecture consistently achieves higher validity across all datasets, but tends to reduce novelty. Furthermore, using multiple augmentation strategies improved the model's ability to generate chemically diverse and novel compounds, as measured by metrics such as novelty and Fréchet ChemNet Distance (FCD). These findings suggest that contrastive learning can offer measurable benefits in de novo molecule generation, although its effectiveness may depend heavily on architecture and dataset-specific tuning.

Keywords: Artificial Intelligence, AI, Deep Learning, DL, Machine Learning, Computer Science, Computer Engineering, Contrastive learning, Self-supervised learning, Representation learning, Data augmentation, Embeddings, Latent space, Generative models, Recurrent neural networks, RNN, Long Short-Term Memory, LSTM, Neural architecture design, Hyperparameter tuning, Transfer learning, Reinforcement learning, NT-Xent loss, Negative log-likelihood, Benchmarking, PCA, T-SNE, UMAP, Drug discovery, De novo molecular generation, In silico screening, Molecular design, Molecular representations, Molecular fingerprints, SMILES, SMILES enumeration, SMILES randomization, Subgraph sampling, Canonicalization, Chemical space, Physicochemical properties, Tanimoto similarity, Fréchet ChemNet Distance, FCD, Synthetic accessibility, SA, Quantitative Estimate of Drug-likeness, QED, Stereoisomers, Stereochemistry, Stereocenters, Tautomerism, Validity, Novelty, Diversity, Internal diversity, IntDiv, ChEMBL35, MOSES benchmark, GuacaMol benchmark, Pharmaceutical AI, Cheminformatics, AstraZeneca, REINVENT, Bioactivity prediction

Acknowledgements

We would like to express our sincere gratitude to Hannes Löffler for his exceptional support and guidance throughout this thesis. His expertise, constructive feedback, and consistent encouragement have been invaluable to the development of our work.

We further recognize examiner Jean-Philippe Bernardy for his valuable feedback and guidance during the work on the thesis, keeping our focus in the right direction.

We also wish to thank Jiazhen He for her valuable insights and the time she dedicated to supervising our work.

Additionally, we acknowledge Simon Olsson for his role as academic supervisor during the project.

Finally, we would like to acknowledge AstraZeneca for providing the opportunity and resources to explore this research topic in an industrial setting.

Hannes Löffler, Gothenburg, 2025-06-19
Jean-Philippe Bernardy, Gothenburg, 2025-06-19
Jiazhen He, Gothenburg, 2025-06-19
Simon Olsson, Gothenburg, 2025-06-19

Abbreviations & Keywords

AI - Artificial Intelligence

BLAS - Basic Linear Algebra Subprograms

BPTT - Backpropagation Through Time

CL - Contrastive Learning

CNN - Convolutional Neural Network

CPU - Central Processing Unit

FCD - Fréchet ChemNet Distance (metric comparing distributions of molecular representations)

GPU - Graphics Processing Unit

GuacaMol - Benchmarking suite for de novo molecular generation based on goal-directed tasks

logP/slogP - Logarithm of the partition coefficient between octanol and water (slogP = simulated logP)

LSTM - Long Short-Term Memory

MOSES - Molecular Sets (benchmarking platform for generative models of molecules)

MW - Molecular Weight

NLL - Negative Log Likelihood

NLP - Natural Language Processing

NT-Xent - Normalized Temperature-scaled Cross Entropy (loss function used in contrastive learning)

PCA - Principal Component Analysis

QED - Quantitative Estimate of Drug-likeness

RL - Reinforcement Learning

RNN - Recurrent Neural Network

SA - Synthetic Accessibility

SGD - Stochastic Gradient Descent

SMILES - Simplified Molecular Input Line Entry System

TL - Transfer Learning

T-SNE - T-distributed Stochastic Neighbor Embedding

TPSA - Topological Polar Surface Area

U-MAP - Uniform Manifold Approximation and Projection

Canonicalization - The process of converting multiple valid SMILES strings of the same molecule into a unique, standardized form. For example, both `C(C)O` and `CCO` are valid for ethanol, but canonicalization ensures only one is used.

Chemical Space - A vast multidimensional set that encompasses possible chemical compounds, often visualized in 2D or 3D plots.

Contrastive Learning - A machine learning technique that trains a model to recognize similar and dissimilar data by comparing augmented views. For example, two enumerated SMILES of the same molecule should be close in embedding space.

Data Augmentation - A method of synthetically expanding the dataset by altering inputs without changing their meaning. In this work, SMILES enumeration or graph-based subgraphing generates new, equivalent molecular views.

De-novo - A Latin term meaning "from the beginning," used in chemistry and computational biology to describe the generation or synthesis of entities from scratch rather than modification. In this thesis, **de novo molecular generation** refers to the creation of entirely new chemical structures by generative models such as REINVENT, without relying on predefined templates or substructures. For example, given only the rules of valid SMILES syntax and a target property, a model might generate a molecule not found in any existing chemical database.

Embedding - A learned numerical representation of input data in a lower-dimensional space. In this project, SMILES strings are transformed into embeddings used for training with contrastive loss.

Fingerprints - A binary vector that encodes structural features of a molecule. Each bit represents the presence or absence of a particular substructure. For example, fingerprints are used to compute Tanimoto similarity between molecules.

Generative Model - A model that learns the underlying distribution of a dataset to generate new, similar samples. In this thesis, an RNN generates novel SMILES strings representing chemically valid molecules.

Latent Space - A compressed, continuous vector space where similar inputs are mapped close together. For instance, contrastive learning embeds different SMILES of the same molecule nearby in latent space.

Novel molecule - Previously unsynthesized chemical compounds with unique properties, in our case – a generated molecule not present in dataset.

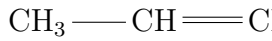
Physicochemical properties - Quantifiable molecular attributes used to evaluate compounds, such as molecular weight, logP, hydrogen bond donors/acceptors, and polar surface area. For example, logP reflects a molecule's hydrophobicity and is essential in drug-likeness assessment.

REINVENT - AstraZenecas in-house generative model for de novo molecular design.

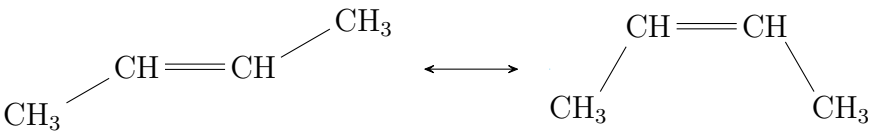
SMILES - Simplified Molecular Input Line Entry System: A textual representation of molecular structure. For instance, the SMILES string for ethanol is `CCO`, representing a chain of carbon-carbon-oxygen atoms.

Stereocenters - Atoms, typically carbon, bonded to four different groups such that swapping any two groups yields a new stereoisomer. For instance, the carbon in lactic acid (CH_3-CH(OH)-COOH) is a stereocenter due to its four distinct substituents.

Stereoisomers - Molecules that share the same molecular formula and connectivity but differ in the three-dimensional spatial arrangement of their atoms. This difference can lead to distinct physical and chemical properties even though the atoms are bonded in the same order. An example of stereoisomerism is the cis-trans isomerism in 2-butene:



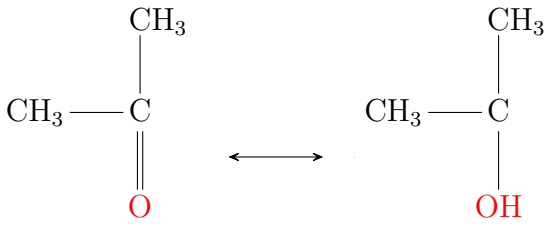
(2-butene)



The left structure represents the "cis" form where both methyl (CH_3) groups are on the same side of the double bond, while the right structure represents the "trans" form where the methyl groups are on opposite sides.

Tanimoto Similarity - A metric for comparing molecular fingerprints, defined as the size of the intersection divided by the size of the union of two sets. For example, two molecules with similar substructures will have a Tanimoto score close to 1.

Tautomerism - A chemical phenomenon where molecules with the same overall formula can shift certain atoms, such as hydrogen, to different positions within the molecule while maintaining the main structure. Specifically, acetone can tautomerize to its enol form, isopropenol:



Acetone
(Keto Form)

Propen-2-ol
(Enol Form)

Contents

List of Figures

List of Tables

1

Introduction

Generative AI has emerged as a transformative tool in both the discovery and design of novel molecules, offering sophisticated capabilities to navigate vast chemical spaces. The term "novel molecules" here refers to previously unsynthesized chemical compounds with unique properties, while "vast chemical spaces" describes the immense variety of possible chemical compounds that traditional approaches often find difficult to fully explore. REINVENT is AstraZeneca's cutting-edge tool for the design of new molecules, working at an atomic level. By harnessing advanced techniques like reinforcement learning and transfer learning, alongside specialized scoring functions, REINVENT is capable of guiding the creation of compounds tailored to achieve desired chemical and biological characteristics[1], [2]. Using Simplified Molecular Input Line Entry Specification (SMILES) strings as molecular representations, REINVENT takes advantage of Long-Short Term Memory networks (LSTM), a type of Recurrent Neural Network (RNN) to train models on large datasets of SMILES. These models are later able to generate chemically valid and diverse compounds as interpretable SMILES strings [3].

A chemical compound can be represented by several valid SMILES strings, which poses a challenge for REINVENT's generative tasks. Since the training process does not include instruction on chemical equivalence, the generated chemical space often contains many redundant compounds. The seminal work by Qian, Shi, and Zhang [4] demonstrates that explicitly teaching equivalence correlates with enhanced novelty in the generated sets. In light of this, developing learning strategies that improve a model's understanding of equivalence is highly desirable. Although the approach in Qian, Shi, and Zhang [4] employed a transformer-based architecture, this thesis investigates whether incorporating equivalence teaching can also benefit models based on LSTMs.

1.1 What's Already Been Built

REINVENT employs a RNN architecture utilizing Long Short-Term Memory (LSTM) cells to capture dependencies within SMILES sequences in distinct time steps. Training is conducted on extensive SMILES datasets using teacher-forcing–a method wherein, during training, the actual target token from the dataset is fed into the next time step instead of the model's own prediction [5]. During training, a probability of generating a specific token at each time step is drawn. By minimizing the negative

log likelihood of such a joint probability, model weights are updated such that the model understands the SMILES structure and syntax. After training, given a token, a probability distribution is derived and a subsequent token is chosen randomly from this distribution. This task can be framed as a sequence-to-sequence generation problem, where a model is trained on sequence-based data-namely, SMILES strings-and generates corresponding SMILES sequences as output. The training process unfolds over multiple phases: the initial step produces a base model, followed by stages that apply transfer learning (TL) and reinforcement learning (RL) to fine-tune the model toward drug-specific attributes. It is during the transfer learning phase that the teacher-forcing method is applied, and this thesis aims to enhance this aspect of the training. Currently, no strategy within this framework explicitly enforces the recognition of chemical equivalence among different SMILES representations, which is what this thesis intends to implement into the sequence generational task.

1.2 Contrastive Learning

Contrastive learning has gained substantial attention in recent years as a powerful technique for representation learning, especially in the domains of computer vision and natural language processing (NLP). This approach trains models to recognize similarities and differences between data points, effectively capturing the underlying structure of the data [6].

One of the seminal works in this domain is SimCLR, proposed by Chen, Kornblith, Norouzi, *et al.* [6]. SimCLR employs a simple yet effective framework that leverages large batch sizes and extensive data augmentation to enhance the quality of learned representations. The model contrasts positive pairs of augmented data with negative pairs in the batch, using the NT-Xent loss to maximize agreement over differing views (See 2.5.2).

Another noteworthy model is MoCo, introduced by He, Fan, Wu, *et al.* [7]. MoCo addresses the challenge of large batch size requirements in contrastive learning by employing a momentum-updated encoder. This mechanism facilitates a dynamic dictionary for negative sampling over many iterations, thereby achieving stable and effective unsupervised learning performance that rivals supervised approaches on various benchmarks.

In addition, BYOL (Bootstrap Your Own Latent), proposed by Grill, Strub, Altché, *et al.* [8], marks a significant departure from prior contrastive methods by eliminating the need for negative pairs altogether. BYOL uses a momentum encoder to create two distinct yet simultaneously optimized network branches, thereby bootstrapping the learning of powerful representations through self-supervised signal alone.

These advances in contrastive learning illustrate the potential to vastly improve model robustness and efficiency in feature learning, providing high-quality representations that benefit downstream tasks such as classification, detection, and segmentation. Moreover, the adaptability of contrastive learning to other domains highlights its versatility and potential to reduce dependence on expensive labeled datasets, thereby paving the way for more widespread application in various fields.

1.3 Thesis Contributions

With the baseline established and the contrastive learning framework briefly introduced in the previous section, this thesis investigates whether AstraZeneca's in-house SMILES string generative model can achieve enhanced performance in terms of chemical novelty, validity, and diversity by integrating a contrastive learning approach-with various data augmentation methods-into the current TL stage of the REINVENT software, based on a Long-Short Term Memory network. Chemical novelty gauges how innovative the generated molecules are compared to existing compounds, validity ensures that these molecules adhere to chemical rules and are syntactically correct, and diversity assesses the range of distinct chemical structures produced. For a detailed discussion on each of these metrics, please refer to Section 2.7. The research questions that will be investigated and answered by the work of this thesis are:

- **RQ1:** Does a model trained to minimize the negative log likelihood of SMILES strings **and** trained to maximize agreement between positive SMILES representations on a LSTM neural network prove superior to models trained only to minimize the negative log likelihood on LSTM neural networks for SMILES string generation in the REINVENT platform, in generating SMILES sets with greater novelty, diversity and validity?
- **RQ2:** Does training to maximize agreement between SMILES representations for the SMILES generation task benefit from using more than one method of SMILES augmentation during training, in generating SMILES sets with greater novelty, diversity and validity?

2

Theory

2.1 SMILES: Simplified Molecular Input Line Entry System

The Simplified Molecular Input Line Entry System (SMILES) was originally introduced by D. Weininger[9], and has since been continuously developed. SMILES is a system for encoding molecular graph-based structures into line notation, and an instance of such an encoding is referred to as a SMILES string.

The encoding process for generating a SMILES string can be visualized as a depth-first graph traversal, given some starting node. From the starting node, atoms and substructures are processed and encoded, token by token.

2.1.1 SMILES Notation

Encoding an entire molecule into a SMILES string follows a strict set of chemical rules for notation, hence this section introduces the most fundamental rules for SMILES. An atom is encoded as its own elemental abbreviation, i.e., an oxygen atom encodes into `O`. Hydrogen with the abbreviation `H` is usually omitted, unless it is needed for correctness in chemical meaning: These specific cases will not be covered in this thesis. Chemical bonds are represented by one of `.`, `-`, `=`, `#`, `$`, `:`. Here, `-` represent a single bond between some atom or structure and are usually omitted, meaning both `C-C-O` and `CCO` are valid, equivalent, encodings. Ring formations are encoded arbitrarily by encoding some node in the ring with a numerical label, marking the start and end of the ring with a numerical label, i.e. a ring of six `C` atoms would be encoded into `C1CCCCC1`. Substructures such as branch formations are required to be encoded with parenthesis, `()`, and the connective bond for the branch within them [9]. In figure 2.1 we give an example of what an encoding for the molecule cyclohexene would look like.

2.1.2 Canonicalization

Although SMILES notation enables several valid encodings of the same molecular structure as a result of flexibility in the atom ordering and branching choices, this variability can lead to inconsistencies when chemical compounds are compared to each other. To ensure consistency and coherency, canonicalization can be used

```
C1=CCCCC1
C1CC=CCC1
C1CCC=CC1
C=1CCCCC1
C1C=CCCC1
```

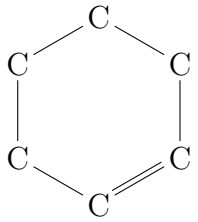


Figure 2.1: Cyclohexene molecule and several valid SMILES encodings. Cyclohexene consists of a closed ring of carbon atoms, denoted by `C1`, with a double bond marked by `=`.

to convert valid SMILES strings of the same chemical compound into a unique, standardized form known as the canonical SMILES. In the continued work of [10], the concept of standardized representations for SMILES and the algorithms to generate them are explained in more detail.

In this study, the datasets were composed of canonicalized SMILES and, in other words: no duplicate data is present in any of the datasets. This approach ensured that the machine learning models were trained on unique and coherent representations of molecules.

2.1.3 SMILES Fingerprints

The fingerprint of a molecule, or *fingerprinting*, is a representation separate from SMILES and a binary representation (bit string) of a molecule. The use case of fingerprints stems from the need to compute similarities between molecules. In the textbook by J. Bajorath [11], each set bit in a vector represents the presence of a specific molecular feature in the compound. Details and listings of these features are omitted in this section.

Assuming a molecule A and molecular features x_k in the range $x_1..n$, the binary-valued vector V_a is computed as

$$V_a = (v_a(x_1), v_a(x_2), \ \dots \ , v_a(x_{n-1}), v_a(x_n))$$

where

$$v_a(x_k) = \begin{cases} 1 & \text{feature is present} \\ b & \text{feature is not present} \end{cases}$$

The length of binary vectors can vary, but are generally $n >> 100$. Throughout this study, molecular fingerprints are 2048 in length.

2.2 Recurrent Neural Networks

As presented by F. Li, J. Johnson and S. Yeung in their lecture at Stanford[12], Recurrent Neural Networks (RNN) are able to solve several upstream and downstream tasks where input data can be interpreted as sequences or where sequential data are desired as output. A first example is the use of convolutional neural networks (CNN) for the upstream extraction of visual features from an image, which are then passed to an RNN to generate a descriptive sentence in a downstream image captioning task, achieved by O. Vinyals et al.[13]. Another example is the work of I. Sutskever et al.[14], where they used an RNN for the upstream encoding of sentences word by word in some language, then decoded by a second RNN for downstream sentence generation in another target language (machine translation). This section introduces the technical- & theoretical description of the RNN and since this work aims to achieve the task of generating SMILES strings, we focus on the task of sequence-to-sequence generation. Throughout this section, terminology and the theory of RNNs are based on the work of I. Goodfellow[15].

RNNs process input sequentially. Given some input vector at some time step and the hidden state, the RNN computes some prediction vector about the next token in the sequence. The key step that is performed to capture temporal information is recurrence, more commonly known as the hidden state h, which at each time step influences future predictions. In later time steps of the RNN, h will contain composed information about all previous time steps. The term *hidden* stems primarily from the fact that this is not a direct input, but rather a value from a previous time step.

We define a sequence x where x_t denotes the embedding vector of the sequence at some time step t, the hidden state at time step t is denoted by h_t (recurrence), and the output values of the RNN block denoted by o_t, the RNN encapsulates the weight matrices of trainable parameters:

$$W_{xh} \in \mathbb{R}^{x \times h}, W_{hh} \in \mathbb{R}^{h \times h}, W_{ho} \in \mathbb{R}^{h \times o}$$

Where W_{xh}, W_{hh} and W_{ho} correspond to the input, hidden, and output matrices, where x is the length of the input sequence, h is the number of hidden units, and o is the output length (note that the length of the input and that of the output are equal). These weights are also shared at every time step t.

The output o_t of the RNN cell is a raw, unnormalized logarithmic probability vector for a specific time step, containing scores for each possible class or token. Then, a vector of token output probabilities, $\hat{y}$, can be obtained by applying a softmax activation function to the output o. The elements of the vector represent the predicted probability distribution for all possible output tokens. This can then be used with a loss function to compare the predicted probability to the ground truth token to train the RNN to increase the predicted probability.

With these definitions, a forward propagation (passing of data through the network) can be defined. The forward propagation calculates the next hidden state and the

output logits. Here, *tanh* is the hyperbolic activation function that compresses any output into the range [-1, 1]. The equations are then as follows:

$$a_t = W_{hh}h_{t-1} + W_{xh}x_t,$$

$$h_t = tanh(a_t),$$

$$o_t = W_{ho}h_t$$

$$\hat{y}_t = softmax(o_t)$$

In the figures 2.2, 2.3, 2.4 we give a brief visualization of the inner workings of the RNN and an example of what processing a SMILES sequence could look like.

With data being sent through a network in incremental time steps, the actual update of network parameters (or weights) W_{xh}, W_{hh} and W_{ho} remains. After forward propagating "left-to-right" through the network, the network's learnable parameters have to be updated. By introducing some loss function L, the previously mentioned vector $\hat{y}$ and the ground truth token vector y, a "right-to-left" parameter update can be defined.

The loss L is calculated at each time step by comparing y to $\hat{y}$, and depending on the loss function measures the distance from how far apart the prediction $\hat{y}$ is from y. However, to minimize the loss and essentially improve future network predictions, the loss gradients L are required to adjust the values of the current parameters. Several details on how the gradients of L is computed are omitted; see [15] for further reading. These gradients are computed with respect to all trainable parameters of the network. Given a learning rate η and loss gradients, the update of the parameters is as follows:

$$\frac{\delta L}{\delta W_{xh}}, W_{xh} \leftarrow W_{xh} - \eta \frac{\delta L}{\delta W_{xh}}$$

$$\frac{\delta L}{\delta W_{hh}}, W_{hh} \leftarrow W_{hh} - \eta \frac{\delta L}{\delta W_{hh}}$$

$$\frac{\delta L}{\delta W_{ho}}, W_{ho} \leftarrow W_{ho} - \eta \frac{\delta L}{\delta W_{ho}}$$

The process just mentioned is referred to as BPTT (back propagation through time) and is the final step in the learning process. This particular instance of update is called stochastic gradient descent (SGD), and by minimizing the loss over each update, the RNN "learns" to make successively better predictions.

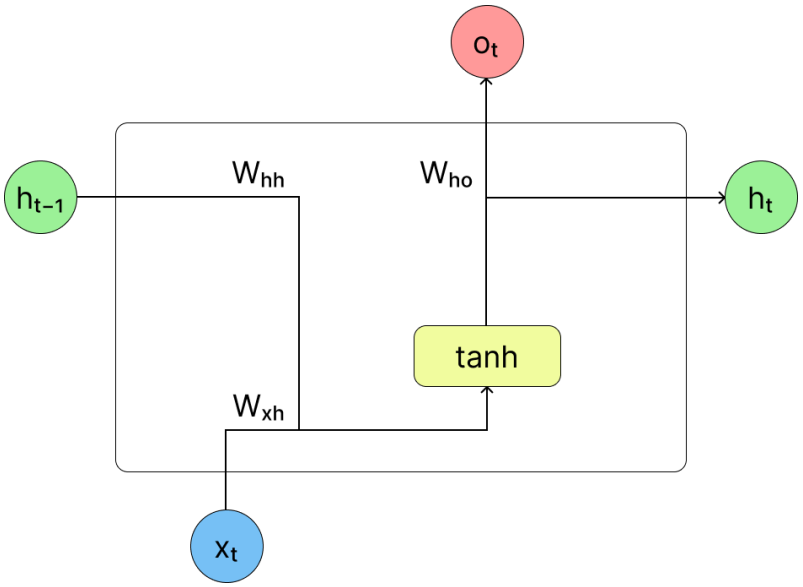


Figure 2.2: Illustration of the RNN cell. At each time step, input vector x_t and previous hidden state h_{t-1} are used to calculate the next hidden state. Applying the weight W_{ho} to h_t yields the output logits o_t that derives the probability distribution $\hat{y}_t$

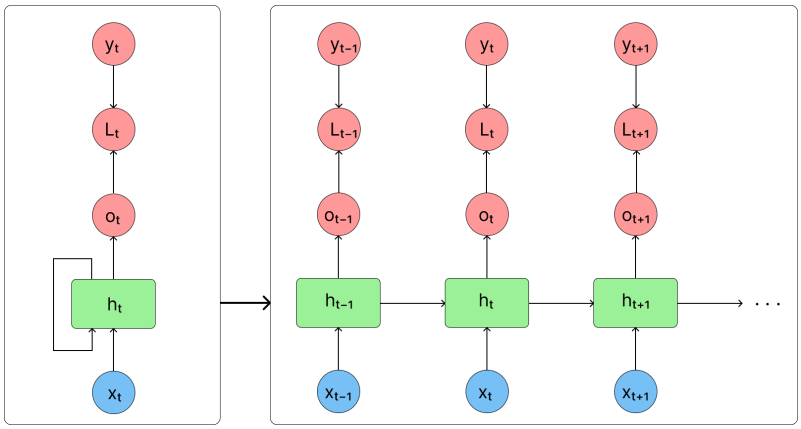


Figure 2.3: Illustration of the RNN unit, and its *unfolded* representation. At each time step t, the next hidden state is computed and *recurred* back into the unit to be used with the next iteration input vector.

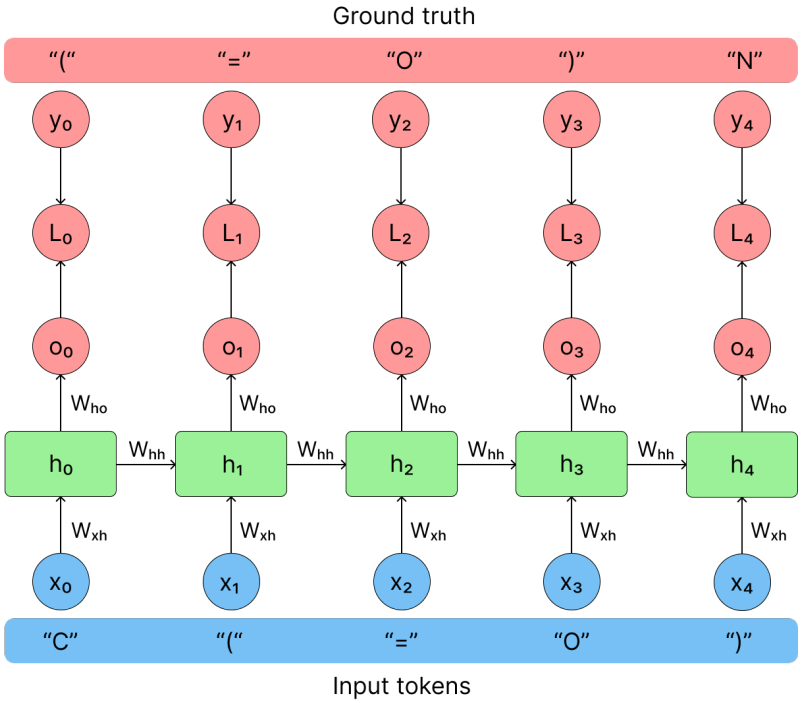


Figure 2.4: Visualized example of forwarding a compound (formamide) in SMILES format through an RNN. In the cases of RNN, the input tokens are encoded as vectors (usually by embedding vectors or one-hot encodings) and passed sequentially to the RNN cell. The output logits are passed through a softmax function to obtain probability vectors of the vocabulary of tokens.

The Vanilla (standard) RNN suffer from one key issue with processing temporal sequences: gradients that accumulate to values either close to zero or numbers of great magnitude. This phenomenon is referred to as vanishing and exploding gradients. In accordance with the paper by I. Goodfellow[15], RNNs apply the same

function several times (once per time step). For simplicity, assuming the recurrence step without the activation function $tanh$, the recurrence relation

$$h_t = W^\top h_{t-1}$$

is applied t times, meaning the final hidden state can also be expressed as $h_t = (W^t)^\top h_0$. Essentially, this can be interpreted as the number of times that a weight W is pressed by itself many times. Depending on the scalar values of W, the final weights W_t will, depending on their magnitude, explode or vanish.

2.3 LSTM: Long Short-Term Memory

The critical issue of vanishing and exploding gradients when processing sequential data can lead to either unstable learning if the gradients become too large or produces uncertainty for in which direction the parameters should be updated if gradients are close to zero. This poses a challenge to maintain long-term dependencies. In this section, it is described how LSTM blocks manage the balance between retaining long-term information and incorporating new input.

The notation for computations carried out by the LSTM follows from the vanilla (standard) LSTM block architecture described in the paper by K. Greff[16], except for the hidden state (see Section 2.3.2) notation which we use h_t, and the notation of gradients of loss function which is referred to as $\frac{\delta L}{\delta *}$. In coherence with the practical implementation of this study, peephole connections are not described. Assuming the input vector x^t to an LSTM block at some time step t, the amount of LSTM blocks N, and M be the total number of inputs, the LSTM layer holds weight matrices for recurrence, input and bias:

$$W_z, W_i, W_f, W_o \in \mathbb{R}^{N \times M},\ R_z, R_i, R_f, R_o \in \mathbb{R}^{N \times N},\ b_z, b_i, b_f, b_o \in \mathbb{R}^N$$

Where z, i, f and o denote the input of the block, the input gate, the forget gate, and the output gate. W, R and b are then the corresponding weights for the input, recurrence, and bias. Activation functions are used for each gate and input. The sigmoid function $\sigma(x) = 1/1 + e^{-x}$ truncates the input in range [0,1], while the hyperbolic tangent function, $g(x)$, truncates the input in range [-1, 1]. With these definitions, a forward pass of sequences can then be expressed with the following equations:

$$\tilde{z}_t = W_z x_t + R_z h_{t-1} + b_z, \quad z_t = g(\tilde{z}_t) \quad \text{(Block input)} \tag{2.1}$$

$$\tilde{f}_t = W_f x_t + R_f h_{t-1} + b_f, \quad f_t = \sigma(\tilde{f}_t) \quad \text{(Forget gate)} \tag{2.2}$$

$$\tilde{i}^t = W_i x_t + R_i h_{t-1} + b_i, \quad i_t = \sigma(\tilde{i}_t) \quad \text{(Input gate)} \tag{2.3}$$

$$\tilde{o}_t = W_o x_t + R_o h_{t-1} + b_o \quad o_t = \sigma(\tilde{o}_t) \quad \text{(Output gate activation)} \tag{2.4}$$

$$c_t = z_t \odot i_t + c_{t-1} \odot f_t \quad \text{(Cell state update)} \tag{2.5}$$

$$h_t = h(c_t) \odot o_t \quad \text{(Hidden state output)} \tag{2.6}$$

Then, the hidden output state h_t is recur in the LSTM blocks for the next input x_{t+1} and the actual output of the LSTM. Similarly to the RNN in the previous section, the LSTM undergoes BPTT on all sets of learnable parameters. To obtain the gradient of some loss function L with respect to the weight matrices, the gradients with respect to the gates, the cell state, and the hidden state. If the recurrent network contains multiple LSTM layers, the gradients of the above layers have to be passed down to the current: Δ^t thus is a vector of those gradients. The matrix R needs to be transposed to match matrix multiplication during BPTT, and is therefore now denoted as $R^\top$:

$$\frac{\delta L}{\delta h_t} = \Delta^t + R_z^T \delta z_{t+1} + R_i^T \delta i_{t+1} + R_f^T \delta f_{t+1} + R_o^T \delta o_{t+1}$$

$$\frac{\delta L}{\delta o_t} = \delta h_t \odot tanh(c_t) \odot \sigma'(\tilde{o}_t)$$

$$\frac{\delta L}{\delta c_t} = \delta h_t \odot o_t \odot \tanh'(c_t) + \delta c_{t+1} \odot f_{t+1}$$

$$\frac{\delta L}{\delta f_t} = \delta c_t \odot c_{t-1} \odot \sigma'(\tilde{f}_t)$$

$$\frac{\delta L}{\delta i_t} = \delta c_t \odot z_t \odot \sigma'(\tilde{i}_t)$$

$$\frac{\delta L}{\delta z_t} = \delta c_t \odot i_t \odot g'(\tilde{z}_t)$$

Each delta is then applied to each respective weight matrix to calculate their gradients. $\langle x, y \rangle$ denotes the outer multiplication of vectors x and y, whose dimensions correspond to that of the corresponding matrix W, R and the bias vector b. Symbol $*$ represents the vector of any of the gates and cell input:

$$\frac{\partial L}{\partial W_*} = \sum_{t=0}^{T} \delta *_t \cdot x_t^\top$$

$$\delta W_* = \sum_{t=0}^{T} \langle \delta *_t, x_t \rangle$$

$$\delta R_* = \sum_{t=0}^{T} \langle \delta *_{t+1}, h_t \rangle$$

$$\delta b_* = \sum_{t=0}^{T} \delta *_t$$

2.3.1 LSTM Cell state

To capture dependencies and contextual information, the LSTM incorporates two major states to keep track of old *and* new information. The *cell state* [16] of the LSTM provides memory of the long-term context in the sequence. In equation 2.5, the forget gate $f(t)$ and the input gate i_t are calculated by adding the previous hidden state y_{t-1} and the current input x_t with their respective gates. These gates respectively control the impact of the new candidate cell state z_t and the old cell state c_{t-1}

2.3.2 LSTM Hidden State

The hidden state, h_t, is the final output of the LSTM block at some time step t and is assigned, contrary to the cell state, to capture short-term sequence dependencies [16]. From equation 2.6, the hidden state is controlled by the output gate o_t and the hyperbolic tangent of c_t. With these controls, h_t captures parts of long-term memory while maintaining balance with the immediate memory of the current input x_t.

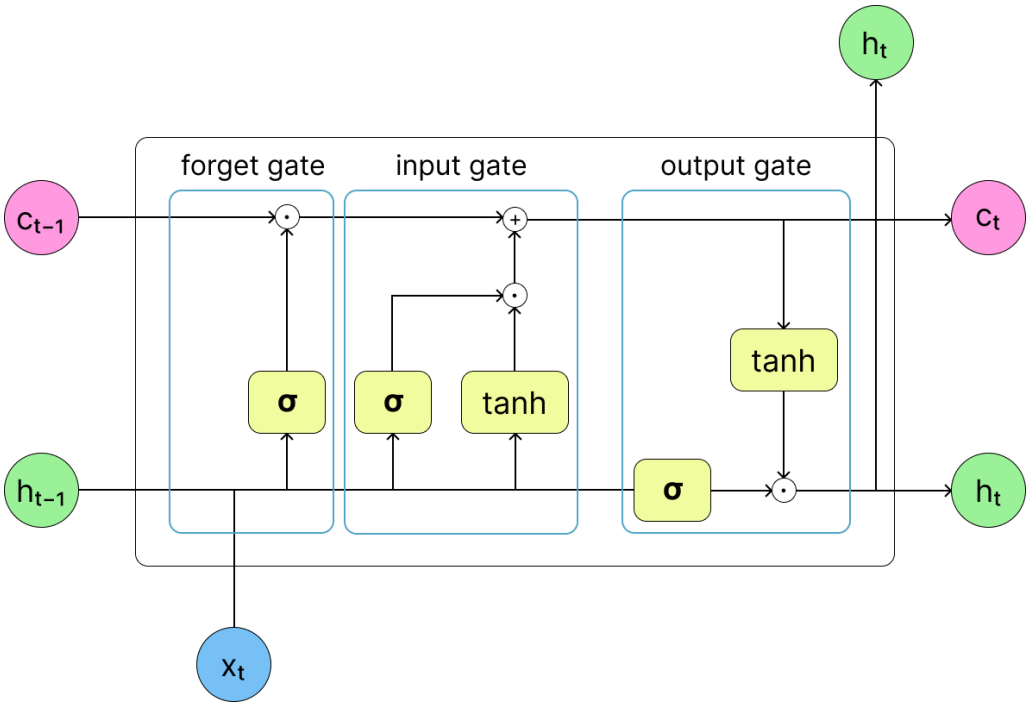


Figure 2.5: Illustration of the LSTM cell, weights omitted from the graphics. Each blue bounding box inside the LSTM cell represent the respective gates. The previous hidden state h_{t-1} is recurred into the LSTM forget, input and output gate. The previous cell state (long-term memory) c_{t-1} recurs back to itself, where it will be considered to be kept depending on the next input x_t

2.4 Data Augmentation

Data augmentation in machine learning refers to the process of creating new training examples by applying transformations to existing data.[17] These transformations generate modified versions that preserve the original label or meaning but introduce variability, enhancing the model's ability to generalize to unseen data. Augmentation is particularly beneficial in scenarios with limited data, as it effectively increases the diversity of the training set without the need for additional data collection.

Figure 2.6 illustrates the concept of data augmentation using an image of a parrot. The original image is transformed into six different versions through various augmentation techniques. These augmentations generate diverse training examples from a single image, enabling the model to learn invariant features and improving its robustness to variations in input data.

Although this study is based on molecular data represented in the SMILES notation rather than images, the principle of data augmentation remains the same. We apply augmentation techniques suitable for chemical data to enhance the diversity of our dataset.

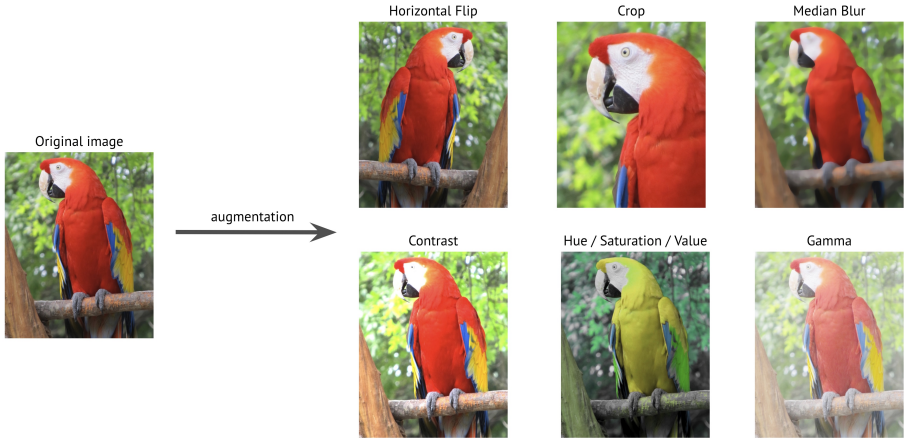


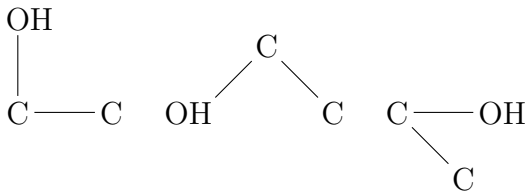
Figure 2.6: Example of Image Augmentations: The original image of a parrot (far left) and six augmented versions applying different transformations: Horizontal Flip, Crop, Median Blur, Contrast Adjustment, Hue/Saturation/Value Shift, and Gamma Correction.

2.4.1 SMILES Enumeration

As mentioned in Section 2.1.1, SMILES is a textual representation of chemical compounds. Each compound has a canonical SMILES string, but many alternative, valid SMILES can be generated due to differences in atom ordering or tautomeric forms. For example, the ethanol molecule can be represented by multiple SMILES strings that encode the same atomic connectivity[18]:

SMILES: `CCO`, `OCC`, `C(C)O`

In graphical form, these molecules look like this:



Although these depictions appear distinct, they all describe the same molecule: ethanol. Although this variability is generally discouraged when constructing datasets for supervised tasks, it aligns well with the principles of contrastive learning. Specifically, enumerating alternative SMILES for a given molecule provides a natural way to generate positive pairs for contrastive augmentation.

2.4.2 Graph Represented Data Augmentation

The molecules graphical representation introduces methods to alter its original structure. Y. You et al.[19] propose various methods of augmentation for graphically represented data. However, if the effects on chemical properties are taken into account when removing and reorganizing atoms and substructures in molecules are taken into account, some methods are found to be more applicable than others. Y. You

et al. found that graph node dropping and subgraphing is beneficial in most data fields, whereas edge perturbation, the procedure of randomly adding, removing or reconnecting chemical bonds starting from the complete molecular graph, could fail to preserve biochemical data.

2.4.2.1 Subgraphing

In appendix A of Y. You et al. [19] work, subgraphing is described as follows: Assume a molecule with nodes and bonds. We denote the graph representation of the molecule by $\mathbf{G} = (\mathbf{V}, \mathbf{E})$ where V and E denote vertices (atoms) and edges (bonds). The construction of a subgraph of the said graph $\mathbf{G}$ is done by a random walk of some length ratio k of the total node size of $\mathbf{G}$.

A starting node is arbitrarily chosen from $\mathbf{G}$. Iteratively, one arbitrarily chooses another node that is adjacent to the starting node. When completed, a subgraph with a set of nodes of size $k|V|$ has been constructed.

2.4.3 SMILES Randomization

The randomization augmentation method follows from SMILES enumeration, and picking one of the enumerated SMILES strings as a positive representation.

2.5 Contrastive Learning

Contrastive learning is a representation learning technique that trains models to recognize similarities and differences among data points. This approach has proven effective in various domains-including computer vision and natural language processing-to enhance model understanding and robustness [6]. For instance, contrastive learning has been successfully deployed in image retrieval, enabling models to differentiate between visually similar objects [20].

At its core, contrastive learning is built on the principle of pulling similar data points (e.g., different augmented views of the same input) closer together in the representation space, while pushing dissimilar data points further apart. Figure 2.7 illustrates this core concept: positive pairs (augmented versions of the same data point) are drawn together, whereas negative pairs (different data points) are pushed apart. Although the figure is shown in a two-dimensional space for clarity, the actual representations lie in a high-dimensional space where each dimension captures unique features. See Section 2.6 for a deeper discussion on visualizing high-dimensional data.

This section is organized as follows. First, we introduce the overall contrastive learning framework- including a discussion on various similarity measures and an explanation of how augmented images are passed through the network (see Figure 2.8). Then, we describe the NT-Xent loss function in detail, including the associated mathematical formulation and gradient dynamics.

2.5.1 Contrastive Learning Framework

A typical contrastive learning framework consists of a neural network that processes input samples to generate embeddings. For each training data point, two distinct augmentations are generated and passed through a shared encoder network to yield two embeddings. The augmentations are generated using various augmentation techniques stated in Section 2.4. These embeddings are then compared using a similarity measure to optimize a contrastive loss function. Figure 2.8 visualizes how augmented images are passed through a network to generate embeddings that will be compared later.

Several similarity measures are commonly used, including:

- **Cosine Similarity**: Measures the cosine of the angle between two vectors, effectively capturing their orientation regardless of magnitude. When embeddings are L2-normalized (i.e., $\|\mathbf{z}\| = 1$), the dot product directly equals the cosine similarity.[6]
- **Euclidean Distance**: Computes the straight-line distance between two points in space, emphasizing the absolute differences in feature values.[21]
- **Dot Product Similarity**: Evaluates the inner product of two embeddings, capturing both magnitude and directional alignment.[15]

In practice, the choice of similarity measure can significantly affect the behavior of the loss function and the quality of the learned representations. For example, cosine similarity tends to be effective in high-dimensional spaces where the direction of the embeddings encodes more informative content than their magnitude.[6]

The similarity measure is designed to maximize the agreement between augmented data points of the same input. Maximizing this agreement is accomplished by engineering a loss function that explicitly encourages high similarity scores (approaching the upper bound, e.g., 1 for cosine similarity) for positive pairs while simultaneously driving the similarity scores of negative pairs toward lower values (or even negative, in some cases)[6].

For instance, when employing cosine similarity between two normalized embeddings $\mathbf{z}_i$ and $\mathbf{z}_j$, the similarity is computed as:

$$s(\mathbf{z}_i, \mathbf{z}_j) = \frac{\mathbf{z}_i \cdot \mathbf{z}_j}{\|\mathbf{z}_i\| \|\mathbf{z}_j\|}.$$

Assuming normalization, this simplifies to:

$$s(\mathbf{z}_i, \mathbf{z}_j) = \mathbf{z}_i \cdot \mathbf{z}_j.$$

A popular formulation used in contrastive learning is the NT-Xent loss. Suppose we have a batch of N original samples, each of which is augmented twice (yielding $2N$ samples). For a positive pair $(\mathbf{z}_i, \mathbf{z}_j)$, the loss is defined as[6]:

$$\mathcal{L}_{i,j} = -\log \frac{\exp\left(s(\mathbf{z}_i, \mathbf{z}_j)/\tau\right)}{\sum_{k=1}^{2N} \mathbf{1}_{[k \neq i]} \exp\left(s(\mathbf{z}_i, \mathbf{z}_k)/\tau\right)} \tag{2.7}$$

where:

- τ is the temperature hyperparameter that controls the sharpness of the resulting probability distribution.
- $\mathbf{1}_{[k \neq i]}$ is an indicator function that excludes comparison of the anchor with itself.

This framework results in embeddings in which different augmentations of the same input are clustered closely, while embeddings from different inputs are well separated. The underlying gradient dynamics ensure that, for positive pairs, the similarity $s(\mathbf{z}_i, \mathbf{z}_j)$ is increased toward its maximum, while for negative pairs, the similarity is correspondingly reduced. Such a structured embedding space captures invariant semantic features and benefits downstream tasks such as classification, clustering, and retrieval[6].

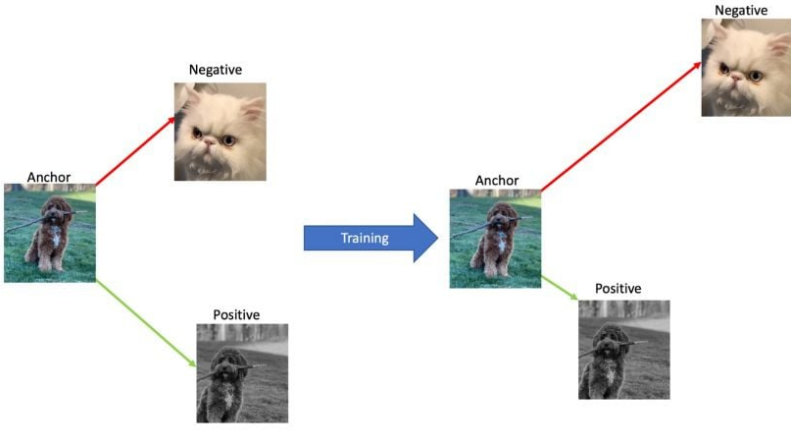


Figure 2.7: Illustration of contrastive learning: Similar images are pulled together, and dissimilar images are pushed apart.

2.5.2 Loss Function: NT-Xent

To effectively learn representations, contrastive learning frequently employs the Normalized Temperature-Scaled Cross Entropy (NT-Xent) Loss, see equation 2.7. The total loss is then calculated as[6]:

$$\mathcal{L} = \frac{1}{N} \sum_{i=1}^{N} \ell_i$$

where (from equation 2.7):

- $\mathbf{z}_i$ and $\mathbf{z}_j$ are the embeddings of a positive pair (i.e., different augmented views of the same input).
- The denominator sums all samples k in the batch (excluding the anchor itself), effectively normalizing the positive similarity against all negative similarities.
- N is the number of anchors (or positive pairs) in the batch, and the final loss is an average over these pairs.

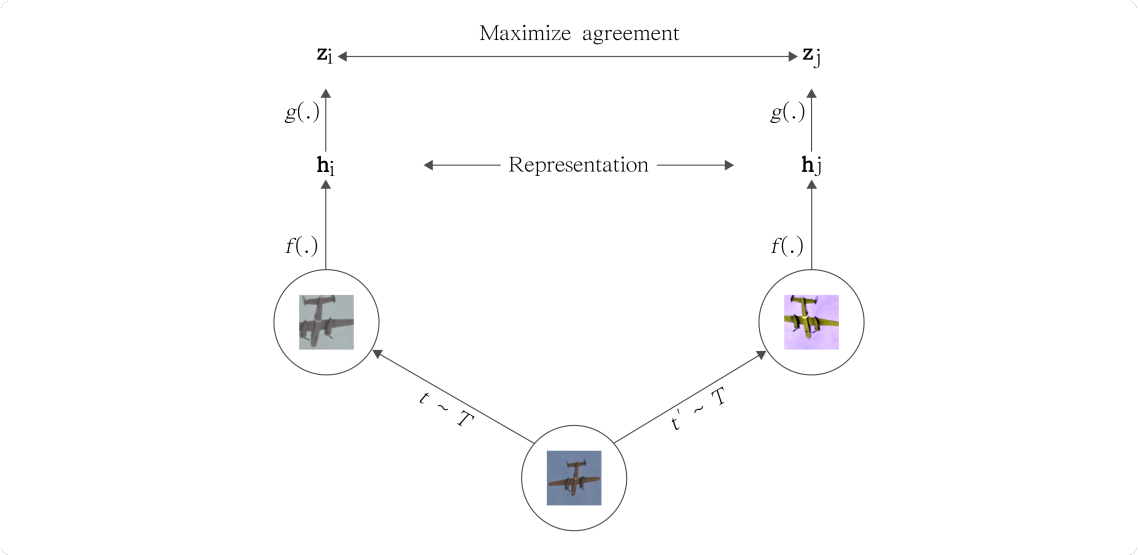


Figure 2.8: Contrastive learning framework: Augmented images are passed through a neural network to generate embeddings for comparison. Image adapted from Thomas St-Hilaire's blog post *Simple Self-Supervised Learning*[22]

In summary, the NT-Xent loss function drives the model to maximize the cosine similarity between positive pairs (pushing the similarity score close to 1) while normalizing these scores across negative pairs to enforce a clear margin of separation. This mechanism yields a highly structured and discriminative embedding space, laying a solid foundation for downstream applications.[6]

2.6 Dimensionality Reduction and Visualization

Getting an idea of how learned representations are structured is a crucial step in evaluating the contrastive learning. Since embeddings exist in a high-dimensional space, interpretation is difficult without dimension reduction[6].

In this section, three widely used methods: T-SNE, U-MAP, and PCA are explored. Each method provides a unique perspective on the underlying structure of the learned embeddings, or chemical space, if you will. This helps assess whether similar molecules are mapped closely together.

While T-SNE and U-MAP are effective in capturing non-linear relationships and preserving local structure, they are primarily designed for visualization rather than data compression. In contrast, PCA applies a linear transformation that highlights variance in the data and can also be used for dimensionality reduction, making it suitable for down-sampling embeddings before further processing.[23], [24]

2.6.1 T-distributed Stochastic Neighbor Embedding (T-SNE)

T-SNE is a non-linear dimensionality reduction technique that is effective at revealing local structure in high-dimensional data. As detailed by Wattenberg et al. in a

seminal paper [25], T-SNE works by converting pairwise similarities in the original space into conditional probabilities, then modeling these in a low-dimensional space using a Student's t-distribution to mitigate the crowding problem. Importantly, the technique is designed to preserve local neighborhood relationships: clusters that appear in a t-SNE plot reliably indicate groups of data points that are similar in high-dimensional space. However, the method does not reliably represent global distances or the relative sizes of clusters, and can sometimes exaggerate the separation between clusters.

From a T-SNE visualization, one can clearly see the formation of distinct clusters or centroids, if you will, that suggest local groupings in the chemical space. These centroids can be interpreted as groups of molecules with similar representations, though caution should be taken not to overinterpret the distances between clusters. The Distill article [25] provides a thorough discussion of both the capabilities and limitations of t-SNE, emphasizing that while it is an excellent tool for visualizing local structure, its depiction of global structure should be viewed with care.

2.6.2 Uniform Manifold Approximation and Projection (UMAP)

UMAP is a non-linear dimensionality reduction technique designed to preserve both local and global structures within high-dimensional data. Introduced by McInnes et al. [26], UMAP constructs a high-dimensional graph representation of the data and optimizes a low-dimensional embedding using a fuzzy topological framework. Unlike T-SNE, which focuses primarily on local relationships, UMAP is better at maintaining some aspects of global structure while still effectively clustering similar points.

UMAP is computationally more efficient than T-SNE and scales well to large datasets. In the context of chemical space visualization, UMAP helps reveal structural similarities among molecular embeddings while providing a clearer representation of the overall dataset structure. However, like all dimensionality reduction techniques, the choice of hyperparameters can significantly impact the resulting visualization, and interpretations should be made cautiously.

2.6.3 Principal Component Analysis (PCA)

PCA is a linear dimensionality reduction technique that transforms high-dimensional data into a lower-dimensional space while preserving the directions of maximum variance. Originally introduced by Pearson [24] and later expanded by Hotelling [23], PCA computes an orthogonal basis of principal components by performing an eigenvalue decomposition of the data covariance matrix or applying Singular Value Decomposition (SVD).

Unlike T-SNE and UMAP, which focus on non-linear transformations for visualization, PCA provides a deterministic, interpretable projection that maintains global structure. One key advantage of PCA is its ability to serve as a down-sampling technique, reducing the dimensionality of embeddings before further processing while

retaining as much variance as possible. This makes PCA particularly useful for preprocessing molecular representations before contrastive learning or clustering analyses.

However, PCA assumes linear relationships in the data and may not effectively capture complex, nonlinear structures that T-SNE and UMAP can reveal. Despite this limitation, it remains a powerful tool for both exploratory analysis and feature reduction in high-dimensional datasets.

2.7 Evaluation Metrics

De-novo generational models for SMILES strings can be assessed by inspecting several metric scores, and the following metrics have been previously used to assess de-novo generation performance in CONSMI[4], GuacaMol[27] and MOSES[28].

A core metric for assessing whether the model has learned the syntax and structure of the SMILES system, the validity metric assesses the fraction of valid SMILES strings given the total sampled output set [4]. This value ranges between 0 and 1, where a score close to 1 indicates that the model recognizes the structure of SMILES well. The computation is straight forward:

$$Validity = \frac{amount\ of\ valid\ SMILES}{amount\ of\ generated\ SMILES}$$

Another important metric to consider for evaluating the generative performance is *uniqueness* [4]. Uniqueness considers a sampled set from the model and counts the total number of samples and the total amount of unique samples. That is, sampled SMILES strings might be enumerations of the canonical SMILES of some molecule. A denoted unique SMILES string is also valid. This metric ranges between 0 and 1.

$$Uniqueness = \frac{amount\ unique\ SMILES}{amount\ of\ valid\ SMILES}$$

Novelty [4] describes the fraction of sampled SMILES strings that are not present in the training set and is represented in the range of 0 and 1. A value close to 1 suggests that the model has a higher capacity for generalization and exploration. However, if this metric is closer to 0, it would suggest overfitting the training data. Denote the amount of unique sampled SMILES as G and the training set of SMILES as T, then:

$$Novelty = \frac{G \setminus T}{G}$$

Internal diversity ($IntDiv_p$) [4] is a metric that calculates the diversity of generated structures and essentially measures how well the model generates nonsimilar structures. This metric is use case specific for SMILES, and relies on measuring the mean

power, p, of tanimoto similarity (see section 2.7.1.1) between pairs of molecular fingerprints. If G is the generated set and $T(s_1,\ s_2)$ is the tanimoto similarity between a pair of SMILES represented as molecular fingerprints s_1 and s_2, then,

$$IntDiv_p = 1 - p\sqrt{\frac{1}{|S|^2}\sum_{s1,\ s2 \in S} T(s1,\ s2)^p}$$

Fréchet ChemNet Distance (FCD) proposed by Preuer et al. [29] is a well-suited metric for comparisons of generative de-novo molecule models. The metric is centralized around their proposed ChemNet neural network, an LSTM based RNN trained to predict biological activity in approximately 6000 assays: experimental procedures to assess the amount and/or quality of some target entity. The assays are common in the well established datasets PubChem [30], ZINC [31], and ChEMBL [32]. Calculating FCD between a generated SMILES set and real-world molecules, FCD essentially captures diversity, as well as the chemical and biological significance of sampled SMILES strings, where a score close to 0 signifies great structural molecular similarity. According to the study, a set size of 5000 samples is sufficient to estimate the means and covariances.

The activations of the penultimate layer of ChemNet are used to obtain a numerical representation of the SMILES molecules. Assuming a sampled dataset from a generative model and a dataset of real-world molecules, their distribution, mean, and covariance are denoted $p(.)$, (m, C) and $p_w(.)$, (m_w, C_w) respectively and Tr denotes the trace of a matrix.

The notation and equation for calculating the FCD is analogous to the one presented in [29].

$$d^2((m, C), (m_w, C_w)) = ||m - m_w||_2^2 + Tr(C + C_w - 2(CC_w)^{1/2})$$

2.7.1 Molecular Properties And Metric Computations

In this section we give a brief introduction to additional compound property metrics, as well as molecular structure properties for completeness.

2.7.1.1 Tanimoto Similarity

There are several measures to score how similar two compounds are to each other. In the textbook by J. Bajorath [11], the tanimoto similarity measure is discussed. Tanimoto similarity depends on fingerprinting molecules (see 2.1.3). Treating a fingerprint of a molecule as a set of present features encoded as a binary vector, the tanimoto similarity computes the intersection and difference to derive a value of symmetric similarity between the molecules. Assuming the fingerprint A and B of two molecules,

$$|A \cap B| = a = \text{amount of features common to } A \text{ and } B$$

$$|A - B| = b = \text{amount of features in } A \text{ but not in } B$$
$$|B - A| = c = \text{amount of features in } B \text{ but not in } A$$

and finally the tanimoto similarity is computed as,

$$S_{tan}(A, B) = \frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{|A - B| \ + |B - A| \ + |A \cap B|} = \frac{a}{a \ + b \ + c}$$

and resides in the range of $0 \leq S_{tan}(A, B) \leq 1$.

2.7.1.2 logP

logP, or the coefficient of octanol / water partition, is a measure of how a molecule distributes itself over a hydrophobic and hydrophilic phase and is explained in the work of Bhal[33]. The metric is widely used in pharmaceutical fields to understand the behavior of drug compounds and is considered to select possible drug candidates. Depending on the intended drug behavior and the use case, different ranges of logP value are sought. Throughout this work, the term logP is interchangeable with xlogP, which in itself is an algorithm to compute the logP value of a compound proposed by Wang[34].

2.7.1.3 Molecular Stereo-centers & -isomers

Stereo-isomers are compounds where "the atoms have the same connectivity but are arranged differently" [[35], 2012, p. 306]. This difference in spatial arrangement is due to the presence of stereo-centers, atoms at which the interchange of two groups results in different stereo-isomers.

3

Methods

3.1 Datasets & Pre-processing

Three datasets were used in the evaluation of the contrastive learning framework. GuacaMol [27], ChEMBL35 and MOSES. The framework for loading datasets was already in place, requiring nothing more than the presence of the datasets on the local machine that runs the contrastive learning training.

A baseline dataset was derived from ChEMBL35 with a filter to obtain drug-like molecules. The filter excludes samples containing atoms B and P and a molecule weight no more than 1200 Daltons. Isotope molecules are included, while molecules with stereo centers are completely removed. What remains is a dataset that contains 2.2M samples. The train, validation and test set split is similar to that of MOSES, with 90% allocated for training, and 5% for validation and testing, respectively.

BenevolentAI [27] proposes a framework for generative de-novo models, aiming to serve as a fair benchmarking platform with a common dataset, GuacaMol. GuacaMol is a subset of the ChEMBL dataset, which contains 1.59M SMILES samples (1.27M train, 0.079M validation, 0.238M test). The benchmark asserts performance for the novelty, validity, and diversity metrics, among others.

Another similar benchmarking tool and dataset is MOSES[28]. MOSES contains roughly 1.94M samples, split into 1.6M, 176K, 176K for training, test, and test scaffold sets, respectively. The set is derived from the ZINC [36] data set, containing molecules with a weight between 250 to 350 Daltons. with a XlogP less than or equal to 3.5. The MOSES set is further filtered to exclude molecules with charged atoms and no other atoms other than C, N, S, O, F, Cl, BR and H. Much like GuacaMol, the benchmark tool assesses validity, novelty, and uniqueness but excludes diversity. See Section **??** for the full description and usage of the benchmarking suites in this thesis.

3.2 Contrastive Learning in LSTM networks

In the setting of LSTM architectures, contrastive learning introduces a framework that enriches the model's representational capacity by training it to distinguish between different augmented views of the same input. This is particularly impactful in

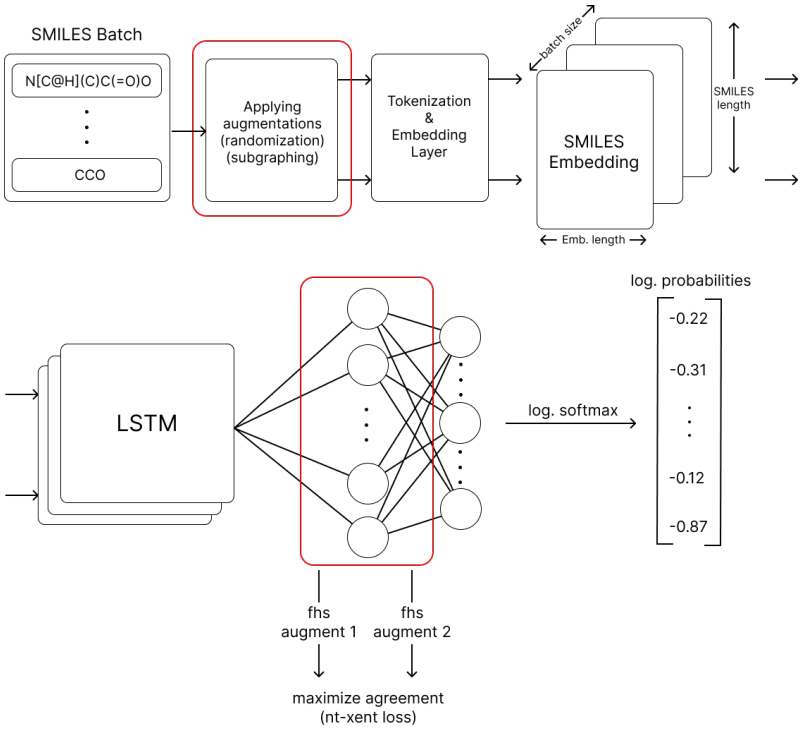


Figure 3.1: Illustration of the REINVENT architecture extended with a contrastive learning branch. Augmented SMILES inputs are encoded via an LSTM network, and the final hidden state h_t is used for computing the NT-Xent contrastive loss.

molecular generative tasks, where SMILES strings can represent the same compound through multiple syntactic permutations.

To enable meaningful comparisons between samples during contrastive learning, the choice of representation extracted from the recurrent network is critical. The LSTMs final hidden state h_t, detailed in Section 2.3.2, captures a compressed summary of the input sequence and its temporal dependencies. This state serves as the anchor point in embedding space for computing similarity across augmented instances.

The process begins by applying augmentations such as SMILES randomization and subgraph sampling (Figure 3.1) to produce distinct yet chemically equivalent inputs. These are then passed through the tokenization and embedding layers, followed by an LSTM encoder. Crucially, the contrastive loss (NT-Xent loss as introduced in Section 2.5.2) is computed on the final hidden state h_t, which acts as a learned embedding of the molecule.

3.3 Model Architecture

The existing model consists of an embedding layer, an LSTM-based RNN encoder, and a linear layer. This corresponds to the architecture shown in Figure 3.2. It processes input through these layers and computes the NLL loss.[3]

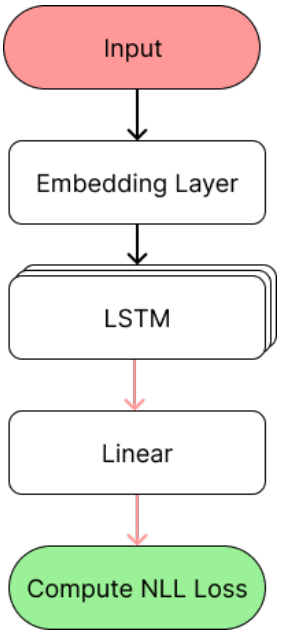


Figure 3.2: Original model architecture. The red arrows indicate the output vector flow. Green blocks denote computations.

Intuitively, the loss of NT-Xent may be applied to the output of the last linear layer; however, a more effective approach is to apply it to the final hidden state of the LSTM. The last hidden state serves as a condensed representation of the input sequence, capturing both local and long-range dependencies learned throughout the stacked LSTM layers[37][6].

However, the last hidden state exists in a high-dimensional space (see Section 2.6), making it inherently noisy and challenging to extract meaningful information. To mitigate this, we propose four modifications to enhance the model's ability to learn better representations in the following subsections.

3.3.1 Architecture: PCA

Instead of adding a layer to the neural network, PCA (more about PCA can be read in Section 2.6.3) could be used to downproject the last hidden state before computing the loss of NT-Xent. This approach leverages PCA's ability to retain the most important variance in the data while reducing dimensionality. The key advantage of PCA in this context is that its computations are deterministic, unlike nonlinear alternatives such as T-SNE, which are stochastic and do not generalize to new data. Since PCA does not introduce any learnable parameters to the network in comparison to a fully connected layer, no backpropagation takes place through this medium and is instead directly computed at the recent LSTM layer with the dimensionality-reduced final hidden state output.

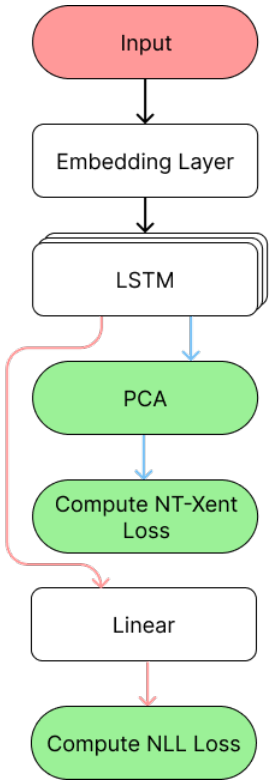


Figure 3.3: Model incorporating PCA for down-sampling before NT-Xent loss computation. The red arrows indicate the output features of the LSTM, while the blue arrows represent the last hidden state output of the LSTM. Green blocks denote computations outside of the network.

3.3.2 Architecture: Intermediate

In this approach, a linear layer is added after the LSTM to project the final hidden state into a lower-dimensional space before applying the contrastive loss. This serves as a flexible and easily learnable downprojection mechanism. By passing the final hidden state through this additional layer, the model is encouraged to encode both syntactic structure and similarity.

3.3.3 Architecture: Existing

Instead of adding a new linear layer, this approach repurposes the existing linear layer to perform down-projection before computing contrastive loss, while still maintaining its original role in processing the final output for NLL loss. This modification simplifies the architecture while effectively incorporating contrastive learning.

3.3.4 Architecture: Intermediate-extended

In this model architecture, we experimented with introducing an additional linear layer alongside the existing one to compute contrastive loss. As in the intermediate architecture, the final hidden state from the RNN is passed through both the new and existing linear layers, in hopes of letting the model learn richer representations

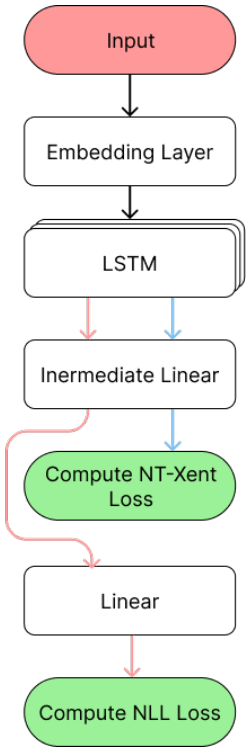


Figure 3.4: Model using an intermediate layer for down-sampling before NT-Xent loss computation. The red arrows indicate the output vector flow, while the blue arrows represent the last hidden state flow. Green blocks denote computations outside of the network.

before the loss is computed.

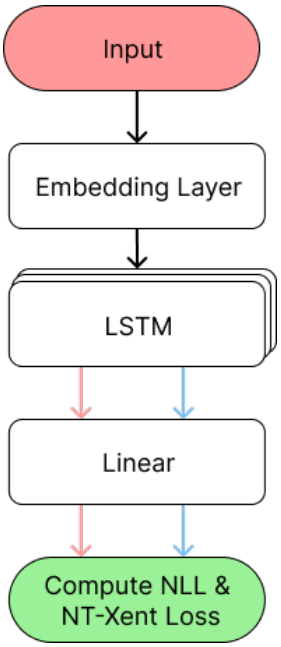


Figure 3.5: Model using the same linear layer for down-sampling before computing both NT-Xent and NLL loss. The red arrows indicate the output vector flow, and the blue arrows represent the last hidden state flow. Green blocks denote computations outside of the network.

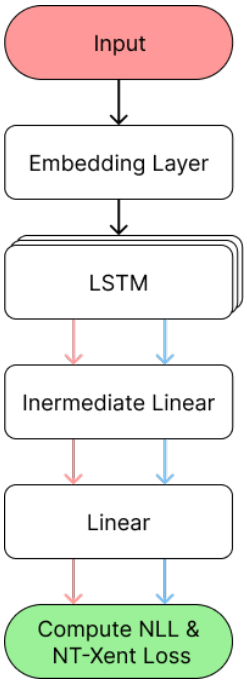


Figure 3.6: Model architecture combining a separate linear layer with the existing linear layer for contrastive loss computation. The red arrows indicate the output vector flow for NLL loss, the blue arrows represent the last hidden state flow for contrastive loss, and green blocks denotes at after which layer in the network losses are calculated.

3.3.5 Architecture: Intermediate-isolated

In this architecture, an additional linear layer was introduced outside of the existing pipeline. This design leverages the downprojection capabilities of a linear layer while ensuring that the performance of the established generative task remains unaffected.

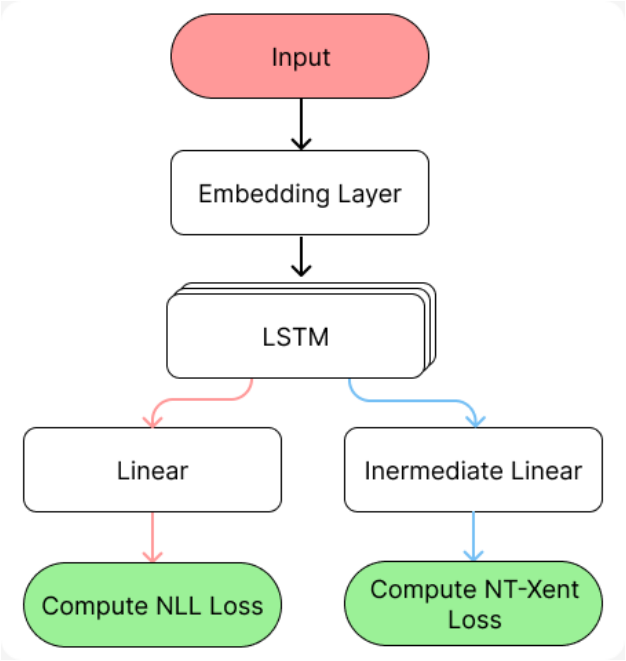


Figure 3.7: Model architecture using two linear layers for different causes. The red arrows indicate the output vector flow for NLL loss, the blue arrows represent the last hidden state flow for contrastive loss, and green blocks denotes at after which layer in the network losses are calculated.

3.4 Data Augmentation

This section describes the implementation of SMILES data augmentation methods. It introduces the required libraries, algorithms, and example implementations.

The RDKit library[38] is used throughout all augmentations. RDKit supports conversions from SMILES strings to molecular objects, which is crucial to maintain chemical validity. RDKit also supports the capability to extract bonds from a set of atoms that can then be used to create a new molecule object. With this functional support, creating a molecule from the subgraphing algorithm is possible.

Since the model is created and trained in separate steps, a saved model after creation contains a set buffer of tokens recognized from the training set. This poses an issue when augmenting SMILES strings, since using RDKit may produce unseen tokens, which at the training stage are not compatible with the software deeming them invalid. The fraction of invalid augments was considered small enough to simply use the original sample during learning instead.

3.4.1 Implementation: SMILES Randomization

The SMILES randomization implementation of the project is based on reordering the atoms of the molecule. A SMILES string is converted to a molecule object, and the number of molecules (excluding hydrogen) is counted. The algorithm creates a list `shuffled_order` of indices ranging from `0` to `num_atoms - 1`. The list is then used with the RDKit function `RenumberAtoms(molecule, newOrder=shuffled_order)`. Below is an example snippet of the implementation:

```
smile = "COC"
molecule = toMolecule(smile)    # conversion to RDKit Mol object

shuffled order = shuffle(range(numAtoms(molecule)))
random molecule = renumberAtoms(molecule, shuffled order)

random smiles = toSmile(random molecule)    # conversion to SMILES
```

3.4.2 Implementation: SMILES Subgraphing

The core algorithm for creating a subgraph of a SMILES string is a random walk. To ensure preserving molecular validity, RDKit was used for molecular manipulation: conversions between SMILES and construction of the subgraph from random chosen bonds from the original sample. An example implementation can be seen in the pseudocode below:

```
smile = "COC"
molecule = toMolecule(smile)    # conversion to RDKit Mol object
k = 0.5     # Fraction of total number of atoms to "walk"

num atoms = numAtoms(molecule)
walk length = min. of (num atoms, num atoms * k)

current atom = random atom in molecule     # index of start atom
visited atoms = {current}

iterate over range(walk_length):
    adjacent atoms = adjacent(current)
    if no adjacent atoms:
        exit iteration
    current = arbitrary atom in adjacent atoms
    add current to visited atoms

# Get bonds between randomly "walked" atoms
visited bonds = getBonds(molecule, visited atoms)

# Construct NEW molecule object from bonds
subgraph = pathToSubMolecule(molecule, visited bonds)

smile subgraph = toSmile(subgraph)
```

3.5 Benchmarking

To validate the effectiveness of our contrastive learning framework, the performance of the models was benchmarked using the evaluation metrics outlined in section 2.7 across the datasets described in section 3.1. The primary objective is to assess whether the contrastive approach improves the quality of generated output relative to the baseline methods.

The best-performing model architecture and training configuration identified in the smaller dataset will be used for comprehensive benchmarking on the datasets from section 3.1. By comparing the performance of this model with that of the original REINVENT baseline (trained under identical conditions), we can rigorously evaluate the success of our implementation of contrastive learning.

Once our implementation has shown success relative to the baseline, we will extend our analysis by comparing our model with other state-of-the-art approaches, such as CONSMI. This additional comparison will further contextualize the performance gains achieved by our contrastive learning framework.

3.5.1 Self-implemented Benchmarking Suite

The REINVENT software integrates with TensorBoard from TensorFlow for enhanced data visualization and analysis. Following completion of a sampling phase, the data is gathered and used to compute various evaluation metrics. These metrics are uploaded as events, which are then processed and displayed by TensorBoard for local browser access and analysis. Initially, REINVENT lacked built-in support for calculating novelty, FCD, internal diversity, and uniqueness, and further showed inconsistencies in validity. Although the software supported the calculation of the metric to some extent, a separate script was created to compute these given metrics.

3.5.2 MOSES Benchmarking Suite

The MOSES benchmarking suite supports a wider variety of metrics. Apart from the metrics presented in section 2.7, MOSES computes Similarity to Nearest Neighbor (SNN), Fragment similarity, and Scaffold similarity. However, this thesis will only discuss and draw conclusions in regard to the already introduced metrics, and the extra results are left to the reader to interpret. The benchmark suite also contains metric computation for a hold-out test scaffold test and, for the same reason as above, results for this set will be disregarded as well.

To use the suite, a clone of the MOSES repository and an installation of RDKit[38] is required. Additional dependencies were installed through the repository setup script `python setup.py install`. Given a list of SMILES, all metrics can be obtained using the following script:

```
from moses.metrics import get_all_metrics

smiles_list = ["CCO", "COC"] # typically read from a generated .csv

```

```
metrics = get_all_metrics(gen=smiles_list, k = [len(smiles_list)])
```

where `smiles_list` is a list of generated SMILES strings and `k` is a list with the amounts of SMILES to be used to compute uniqueness, which was set to the size of the generated set. For a complete MOSES description of the benchmarking suite, see [39].

3.5.3 GuacaMol Benchmarking Suite

The GuacaMol benchmarking suite supports computing validity, uniqueness, novelty, and FCD, but does not include internal diversity. The tool requires installation of the GuacaMol, FCD, and RDKit [38] libraries. However, during the benchmarking phase, it was noticed that the suite was poorly maintained due to deprecated dependencies and source code. It was therefore decided to instead benchmark models trained on the GuacaMol dataset on the self-implemented benchmarking suite. The suite also required sampling from trained models during runtime, which did not integrate well with existing REINVENT software.

3.6 Loss Functions

Our model is trained using two complementary loss functions: NLL loss and contrastive loss of NT-Xent. The NLL loss was originally used in the REINVENT framework to optimize the generation of SMILES sequences by maximizing the log-probability of correct tokens[3]. The loss of NT-Xent is integrated, ensuring that similar molecules remain close to each other in the chemical space. Detailed explanations of both loss functions and their implementations are provided in the following sections.

3.6.1 Negative Log Likelihood Loss

In the original REINVENT implementation, the Negative Log Likelihood (NLL) loss was used as the primary loss function for training. This loss function, implemented from `torch`, computes the negative logarithmic probability of the correct class given the model output. According to the PyTorch documentation [40], for an input tensor $\mathbf{x}$ of size (N, C), where N is the batch size and C is the number of classes, and a target tensor $\mathbf{y}$ of size N with class indices $0 \leq y_i < C$, the unreduced loss ℓ for each sample i is defined as:

$$\ell_i = -w_{y_i} \cdot x_{i,y_i},$$

where x_{i,y_i} is the log-probability of the correct class y_i for the i-th sample, and w_{y_i} is an optional weight for class y_i.

3.6.2 Implementation of NT-Xent loss

As described in Section 2.5, contrastive loss is used to learn an embedding space where different augmentations of the same data point are drawn closer together (positive pairs), while embeddings of different data points are pushed apart (negative

pairs). In our work on molecular representation learning using SMILES strings, we employ the NT-Xent loss to ensure that different augmentations of the same molecule yield similar latent representations.

3.6.2.1 Initial Loop-based Implementation.

Initially, our implementation computed the NT-Xent loss by elementally comparing each pair of augmented sequences using two nested `for` loops. For a batch of N sequences, where each sequence produces two augmented samples, we obtain $2N$ embeddings $\{\mathbf{z}_i\}_{i=1}^{2N}$. The loss for each embedding is calculated as described in equation 2.7, with a slight variation of the numerator here described as $exp(sim(z_i, z_{i+})/\tau)$.

where $\text{sim}(\cdot,\cdot)$ denotes cosine similarity and i^+ indicates the index of the positive pair for i. Since every pair is computed explicitly, this approach has a complexity of $\mathcal{O}((2N)^2 \cdot d)$, with d being the embedding dimension.

3.6.2.2 Optimized Matrix Multiplication Implementation

To overcome the inefficiency of nested loops, we reformulated the computation using matrix multiplication. By concatenating the embeddings from both augmentations, z_1 and z_2, into the matrix

$$\mathbf{Z} = \begin{bmatrix} z_1 \\ z_2 \end{bmatrix} \in \mathbb{R}^{2N \times d},$$

and normalizing each row to obtain $\hat{\mathbf{Z}}$, we compute the complete pairwise cosine similarity matrix in one operation:

$$\mathbf{S} = \frac{\hat{\mathbf{Z}}\,\hat{\mathbf{Z}}^\top}{\tau}.$$

This matrix $\mathbf{S}$ is of size $2N \times 2N$ and contains cosine similarity scores for every pair of embeddings, including comparisons between original data samples, between augmentations, and between a data sample and its augmentation. Note that the diagonal entries of $\mathbf{S}$ represent the self-similarity scores (which are trivially equal to 1, or $1/\tau$ after scaling) and would skew the loss computation if not addressed. To remove their influence, we mask these diagonal elements by setting them to $-\infty$, ensuring that

$$\exp(-\infty) = 0.$$

For the NT-Xent loss, although the numerator uses only the similarity score from the positive pair (i.e., for each sample, the augmented pair is located at index i^+, where for the first N embeddings $i^+ = i + N$, and for the last N embeddings $i^+ = i - N$), the denominator requires summing over all non-self similarity scores. Therefore, the loss for each embedding i is calculated as:

$$\ell_i = -\log \frac{\exp\left(S_{i,i^+}\right)}{\sum_{j=1}^{2N} \exp\left(S_{ij}\right)},$$

and the overall loss is the average over all $2N$ embeddings:

$$L = \frac{1}{2N} \sum_{i=1}^{2N} \ell_i.$$

3.6.2.3 Complexity Consideration

Although both loop-based and matrix multiplication implementations have a theoretical complexity of $\mathcal{O}((2N)^2 \cdot d)$, the optimized approach benefits from highly optimized BLAS (Basic Linear Algebra Subprograms) routines and GPU parallelism. This results in significantly faster execution, especially for large N.

In summary, by reformulating the computation of the NT-Xent loss using matrix multiplication and carefully masking out the self-similarities, we obtain an efficient implementation that scales well with the batch size while preserving the theoretical underpinnings of contrastive learning.

3.6.3 Combining Loss Functions

When testing our new implementation of the NT-Xent loss function, we quickly realized that while the model was learning useful representations for the contrastive learning component, it simultaneously lost its ability to generate chemically valid syntax, that is, the NLL loss increased significantly. This observation led us to combine the loss functions.

To balance the objectives of contrastive learning and maintaining chemical syntax, we adjusted the loss function by dampening the impact of NLL loss while amplifying contrastive loss. Specifically, we reduced the NLL loss by a factor of 10 and increased the contrastive loss by a factor of 10. The modified loss function becomes

$$\mathcal{L} = \frac{1}{10} \cdot \mathcal{L}_{\text{NLL}} + 10 \cdot \mathcal{L}_{\text{Contrastive}} \tag{3.1}$$

3.7 Hyperparameter Tuning

The original network includes batch size, sequence embedding layer size, LSTM layer size, and number of LSTM blocks as non-trainable parameters. Introducing the contrastive learning framework and another fully connected layer further adds intermediate (hidden) fully connected layer size and temperature as parameters applicable for fine-tuning.

The hyperparameter tuning framework Optuna [41] was integrated, and to intain a balanced trade-off between the computational time required for the tuning process and the absence of knowledge of the impact of the parameters on the final performance, the framework was set to conduct a random search in the parameter value space listed in table **??**. Given our four proposed LSTM network architectures and three learning approaches (NLL, CL, $NLL - CL$), 12 hyperparameter tuning runs were performed. The framework was set up to optimize for minimization of the

mean sample loss. The validation set of our 2.2M ChEMBL35 based set was used for tuning, where we selected 22.0K samples from the total of 110.0K samples. Each run of tuning was set for 50 trials of different parameter configurations with 100 epochs each. By carrying out a vast variety of trials, we find not only suitable hyperparameters, but also which architecture excels in our endeavor. This evaluation decides which architecture and hyperparameters will be used for benchmarking.

In order to reduce the heavy computations of finding the best hyperparameters, the batch size will be selected by manually checking how the mean loss varies with different batch sizes. The batch sizes that will be tested are 128, 256 and 512.

Parameter	**Value**				
temperature (τ)	0.05	0.1	0.5	1.0	1.5
embedding size	32	64	128	256	512
LSTM size	64	128	256	512	
No. LSTM blocks	2	3	4		
intermediate size	128				

Table 3.1: Hyperparameters and respective values for tuning

3.8 Experimental Setup

This section introduces the experimental setups that were performed in this thesis. The section introduces settings and the environments for model training and model comparison through benchmarking metrics; crucial for obtain results and eventually conclusions to answer the thesis' research questions.

3.8.1 Experiment: REINVENT & CL framework

To establish a fair evaluation between the original implementation of a REINVENT software-based model and a model based on a contrastive learning framework, the models were trained under the same circumstances to ensure that we can be certain that external factors are not impacting the results of the experiments. For example, a CL model trained with an additional fully connected layer compared with a baseline REINVENT model without such a layer produces a level of uncertainty. One cannot be sure that the contrastive framework or the fully connected layer gave rise to superior or inferior results. To ensure certainty of the impact that the framework has on the results, models would be trained and compared under the same architecture, with the exception of additional hidden layers that were disconnected from the NLL head.

From hyperparameter tuning, it was decided that the architecture that achieved the lowest sample loss for both training strategies was the intermediate architecture (see Section 5.1 for additional information). The models would be trained with the same choice of parameters. Given that NLL loss during training could converge at different rates, it was decided to sample models at the specific epochs where their NLL loss, in other words, training progress, was matching. and For each dataset

mentioned in Section 3.1, a respective model would be trained. The tables below list the said parameters, but exclude the final layer output size since this value depends on the datasets.

Table 3.2: Model configuration: Training and network parameters

Network parameters	
Embedding layer size	256
LSTM Layers	4
LSTM Layer Size	512
Hidden Size	512
Hidden proj. Size	256
Training parameters	
Batch Size	128
Batch Shuffling	None
Learning Rate (η)	0.0001
Number of Epochs	20
Optimizer	Adam
Dropout Rate	0.0

3.8.1.1 Sampling & Scoring

The models were evaluated from their generated sample set. Since certain evaluation metrics such as FCD propose a minimum of 30 000 samples, the choice was made to sample that amount of SMILES strings. The software of REINVENT supports both multinomial and beam search sampling. However, due to the computational requirements of beam search, the multinomial strategy was chosen. The following list presents all parameters involved in the sampling phase.

Table 3.3: Model configuration: Sampling parameters

Network Parameters	
Sampled set size	30 000
Sampling strategy	Multinomial

3.8.2 Experiment: Augmentation Techniques

This section introduces the experimental setup for the evaluation of the augmentation techniques. Several settings are similar to the main experiment in section 3.8.1, however, this experiment was only conducted on the baseline dataset.

The experiment was set up to compare a model trained solely on randomization augmentations and a model that utilized subgraph augmentations on top of randomization augmentations, respectively, referred to as $M_{NLL-CL(R)}$ and $M_{NLL-CL(R,S)}$.

Randomization augmentation is a technique that stems from the variability that the SMILES syntax introduces, while the subgraph technique treats the SMILES string as a graph and extracts a small portion of the original. By choosing two augmentation techniques that consider different aspects of the SMILES representation, the experiment aims to assess how the use of multiple augmentation strategies influences model performance. Due to time constraints, a model trained solely on the subgraphing technique was not tested.

Table 3.4: Model configuration: Training and network parameters

Network parameters	
Embedding layer size	256
LSTM Layers	4
LSTM Layer Size	512
Hidden Layer Size	512
Training parameters	
Batch Size	128
Batch Shuffling	None
Learning Rate (η)	0.0001
Number of Epochs	20
Optimizer	Adam
Dropout Rate	0.0

4

Results

This section covers the results from the thesis' proposed experiments to cover the research questions. The notation for a baseline REINVENT model is denoted as M_{NLL}. Models trained solely under the contrastive framework are denoted as $M_{CL\ (augment)}$, and models trained over the REINVENT framework in combination with contrastive learning are denoted as $M_{NLL-CL\ (augment)}$, where *augment* refers to the augmentation method and can be R (randomization) or S (subgraphing).

4.1 Results From Architectural Tuning

This section covers the results of several tuning trials for the proposed architectures. First, trials were carried out to find the optimal parameters of the network. This was then followed by separate trials in which the parameters of temperature, τ, and batch size parameters were isolated.

4.1.1 Results: Existing Architecture

Table 4.1: Network Architecture and Training Parameters

	$M_{CL\ (R)}$	M_{NLL}	$M_{NLL-CL\ (R)}$
Architecture Parameters			
Num Layers	2	4	2
Layer Size	64	256	256
Intermediate Layer Size	-	-	-
Embedding Layer Size	512	512	512
Sample Loss (Arch)	2.84	36.68	7.5
Training Parameters			
Batch Size	-	64	128
Temperature	-	-	0.05
Sample Loss (Train)	-	36.68	6.99

4.1.2 Results: Intermediate Architecture

Table 4.2: Network Architecture and Training Parameters

	$M_{CL\ (R)}$	M_{NLL}	$M_{NLL-CL\ (R)}$
Architecture Parameters			
Num Layers	2	4	2
Layer Size	256	256	256
Intermediate Layer Size	128	512	512
Embedding Layer Size	512	512	512
Sample Loss (Arch)	2.06	23.29	3.93
Training Parameters			
Batch Size	512	128	128
Temperature	1.0	NA	1.0
Sample Loss (Train)	4.43	29.85	3.87

4.1.3 Results: Intermediate-extended Architecture

Table 4.3: Architecture and Training Metrics for Baseline Dataset

	$M_{CL\ (R)}$	M_{NLL}	$M_{NLL-CL\ (R)}$
Architecture Parameters			
Num Layers	2	4	2
Layer Size	256	256	256
Intermediate Layer Size	128	512	512
Embedding Layer Size	512	512	256
Sample Loss (Arch)	3.14	23.29	6.28
Training Parameters			
Batch Size	-	-	128
Temperature	-	-	0.05
Sample Loss (Train)	-	-	7.44

4.1.4 Results: Intermediate-isolated Architecture

Table 4.4: Architecture and Training Metrics for Baseline Dataset

	M_{CL}	M_{NLL}	M_{NLL-CL}
Architecture Parameters			
Num Layers	2	4	2
Layer Size	128	256	256
Intermediate Layer Size	128	512	256
Embedding Layer Size	512	256	512
Sample Loss (Arch)	2.04	35.80	4.85
Training Parameters			
Batch Size	-	-	128
Temperature	-	-	0.1
Sample Loss (Train)	-	-	4.84

4.2 Contrastive Learning Framework Benchmarks

In this section we will report the results of our model training and report each respective metric. Below we have set up tables to show how we would like to report the metrics. Apart from the tables, we will have graphs of the model's respective loss, and t-SNE, U-MAP and PCA plots to visualize the spread of the high-dimensional data.

4.2.1 Baseline Dataset

Table 4.5: Metrics for baseline dataset across different architectures

	Existing Architecture		**Intermediate Architecture**	
Metric	M_{NLL}	$M_{NLL-CL\ (R)}$	M_{NLL}	$M_{NLL-CL\ (R)}$
Validity	0.884	0.887	0.945	0.935
Novelty	0.985	0.982	0.956	0.962
Uniqueness	0.999	0.999	0.999	0.999
IntDiv_1	0.999	0.999	0.999	0.999
IntDiv_2	0.999	0.999	0.999	0.999
FCD	0.524	0.571	0.334	0.349

4.2.2 MOSES dataset

Table 4.6: MOSES Benchmark Metrics

	Existing architecture		**Intermediate architecture**	
Metric	M_{NLL}	$M_{NLL-CL\ (R)}$	M_{NLL}	$M_{NLL-CL\ (R)}$
Validity	0.913	0.909	0.954	0.971
Uniqueness@1k	1.0	1.0	1.0	1.0
Uniqueness@10k	0.998	0.999	0.996	0.993
Novelty	0.970	0.962	0.937	0.936
Filters Passed	0.977	0.985	0.986	0.993
Frag (test)	0.993	0.992	0.988	0.984
Frag (testSF)	0.991	0.990	0.986	0.985
Scaff (test)	0.790	0.773	0.742	0.559
Scaff (testSF)	0.130	0.169	0.108	0.116
SNN (test)	0.530	0.542	0.560	0.570
SNN (testSF)	0.506	0.519	0.530	0.116
IntDiv_1	0.857	0.850	0.858	0.848
IntDiv_2	0.851	0.844	0.851	0.842
FCD (test)	1.127	1.261	1.342	1.532
FCD (testSF)	1.712	1.789	1.860	1.866

4.2.3 GuacaMol dataset

Table 4.7: Metrics for GuacaMol

	Existing Architecture		**Intermediate Architecture**	
Metric	M_{NLL}	$M_{NLL-CL\ (R)}$	M_{NLL}	$M_{NLL-CL\ (R)}$
Validity	0.840	0.839	0.922	0.917
Novelty	0.987	0.989	0.965	0.969
Uniqueness	0.999	0.999	0.999	0.999
IntDiv_1	0.999	0.999	0.999	0.999
IntDiv_2	0.999	0.999	0.999	0.999
FCD	0.797	0.759	0.370	0.330

4.3 Results on Augmentations

(Here we will report results on the contrastive learning framework and report what/which method(s) of augmentations showed superiority regarding metrics. We will have graphs here as well of the loss and the t-SNE. Below we an example of how we would report this)

Table 4.8: Metrics for baseline dataset: augmentation experiment

Metric	M_{NLL}	$M_{NLL-CL\ (R)}$	$M_{NLL-CL\ (R,S)}$
Validity	0.945	0.935	0.918
Novelty	0.956	0.962	0.973
Uniqueness	0.999	0.999	0.999
IntDiv_1	0.999	0.999	0.999
IntDiv_2	0.999	0.999	0.999
FCD	0.334	0.349	0.311

5

Discussion

5.1 Architectural Tuning Evaluation

After performing hyperparameter tuning, we found that the intermediate architecture yielded the best results. It outperformed the others in terms of sample loss, which we consider to be a suitable metric for the task at hand. However, the models were trained on a relatively small dataset, which differs significantly from the full dataset they will be trained on in the future. This limited data led to high variability in the results, making it challenging to conclusively determine which model or hyperparameter configuration is truly optimal. However, due to time constraints, this approach was the most practical given our current resources.

We are aware that models with alternative hyperparameter settings-beyond what our tuning identified-may perform better. This means that our tuning process is not exhaustive or definitive, but it still provides a valid and useful indication of what configurations are promising. It is also worth noting that we primarily evaluated models based on sample loss, but we could have used validation loss instead, which would also have been a valid approach.

We opted for a random search strategy due to time constraints. Conducting a full grid search across all hyperparameters would have required approximately 900 trials, which could take up to a year to complete without parallelization. Although grid search offers exhaustive coverage, it becomes computationally impractical with a large number of parameters. Fortunately, with support from AstraZeneca, we had access to sufficient computational resources to run 50 random search trials over a few days, allowing us to explore the hyperparameter space efficiently within our time plan.

Based on our findings that the intermediate architecture was the best performing, we used it as one of the foundations for all subsequent experiments.

5.2 Evaluation of Model Trainings

For each dataset, we trained models using both the original (i.e., existing) architecture and the intermediate architecture derived through hyperparameter tuning.

For each architecture, two variants were trained: one using only the NLL loss and another combining the NLL with the CL loss. This experimental design allowed us to assess the impact of architectural modifications on model performance and isolate the effect of incorporating contrastive learning. Interestingly, the results varied notably across datasets-some showed clear improvements with the tuned architecture or CL integration, while others exhibited performance degradation. These unexpected variations highlight potential dataset-specific interactions between model configuration and learning objectives, which we analyze in detail in the following section.

5.2.1 Baseline Dataset Evaluation

The results in the baseline dataset, as shown in Table **??**, demonstrate promising generative capabilities in both architectures. Although both models achieved consistently high scores on uniqueness and internal diversity (IntDiv_1 and IntDiv_2), slight differences in other metrics were observed depending on the architecture and loss configuration.

In the existing architecture, the model trained with the contrastive learning framework (M_{NLL-CL}) achieved a marginally higher validity score (0.887 vs. 0.884) compared to the model only with NLL (M_{NLL}). However, it showed a slight drop in novelty (0.982 vs. 0.985). Interestingly, the FCD (Fréchet ChemNet Distance) score improved with CL (0.571 vs. 0.524), suggesting enhanced chemical relevance or distributional similarity to the training data.

With the intermediate architecture, the contrastive learning model again showed slightly higher novelty (0.962 vs. 0.956), while validity decreased by approximately 1% (0.935 vs. 0.945). As with the existing architecture, FCD improved modestly when using CL (0.349 vs. 0.334). Uniqueness and diversity metrics remained saturated across all configurations.

These findings suggest that, while the incorporation of contrastive learning yields marginal trade-offs across some metrics, it may offer benefits in distributional alignment, as indicated by FCD. However, the effects appear to be architecture-dependent, underlining the importance of tuning the model structure and training objectives in tandem.

5.2.2 MOSES dataset evaluation

In the analysis of the results of the MOSES dataset, the metrics indicate promising generative capabilities across different architectures, with particular highlights observed in the use of the intermediate architecture. Both the existing and intermediate architectures maintained high scores in terms of uniqueness and internal diversity, demonstrating their proficiency in generating a wide range of distinct molecular entities. In particular, the intermediate architecture that uses the contrastive learning framework (M_{NLL-CL}) achieved an especially high validity score of 0.971, surpassing the configuration with only NLL, which recorded a score of 0.954.

This underscores the superiority of contrastive learning in preserving the chemical validity of structures within this architecture.

Despite this improvement in validity, the novelty of the molecules dropped slightly, with the NLL-CL configuration achieving a score of 0.936 compared to 0.937 in the NLL-only model. This highlights the potential trade-off between generating chemically valid structures and exploring novel chemical spaces. Interestingly, the FCD score was worse in both existing and intermediate architectures when the contrastive learning framework was used, suggesting that the generated molecules were less aligned with the distribution of the training data, thus rendering them potentially more irrelevant or dissimilar chemically.

5.2.3 GuacaMol Dataset Evaluation

Unlike the other datasets, the contrastive framework models showed consistently better results compared to the REINVENT framework models under the GuacaMol dataset. The slightly superior novelty and FCD scores across both architectures show some promise of greater generational capabilities. However, the validity was shown to be somewhat lower by 0. 05%, suggesting that contrastive learned models almost captured the SMILES syntax, as well as REINVENT trained models.

5.3 Augmentation Experiment Evaluation

The experiment, which used a combination of randomization and subgraph techniques for data augmentations, produced somewhat more interesting results. Although the model based on this combination achieved lower validity compared to the REINVENT and contrastive model with randomization augmenting, it did prove to be superior by almost 1,7% in novelty, and an FCD score of 0.311 compared to the REINVENT model's score of 0.334.

This discovery could hint that constructing more positive and negative pairs of SMILES samples for each batch is beneficial to the model's capabilities to produce samples that are distinguishable from the original training sets. These findings could also suggest that not only does the number of positive and negative pairs positively impact performance, but when other classes of varied samples are introduced, the performance of novelty increases as well.

5.4 General Evaluation

Examining the results in all datasets, as presented in Table 4.5, Table 4.6, and Table 4.7, there is no conclusive evidence to suggest that the contrastive learning framework consistently outperforms the baseline in any specific metric. Performance varies notably between datasets, raising the question of whether dataset-specific hyperparameter tuning could have yielded different outcomes.

However, what is consistently observable is the superior validity performance of the

intermediate architecture across all datasets and configurations. This gain in validity appears to come at the cost of reduced novelty. Furthermore, the results suggest that the use of multiple augmentation techniques can enhance the overall capacity of the model to generate new compounds and achieve a lower FCD, indicating improved distributional alignment with the reference set.

5.5 Assessment Reliability

A major discussion point is the analysis of the model metrics. For each dataset, architecture, and training strategy, only one iteration of sampling and scoring was carried out. Obtaining results and drawing conclusions on this topic could potentially introduce a degree of uncertainty. Instead, several sampling and metric calculations should be performed to obtain average scores and standard deviations to gain more certainty in the results.

5.6 Unpredicted Behavior & Other Discoveries

A key parameter crucial to analyze during training is the gradient. The loss gradients of the negative log likelihood followed a pattern in which the gradients did not explode nor vanish. However, the gradients of the loss of NT-Xent were incredibly close to zero, even during the very first iterations of forward propagation and BPTT during all trainings. Although after several iterations this behavior seemed to disappear, they were smaller in magnitude in comparison to the gradient of the NLL loss. It is important to be critical when gradients are redundant as it means that the learnable parameters of the network are not affected and updated in a meaningful way. The contrastive framework utilizes weighting of the NLL and NT-Xent losses to maintain a balance of influence from the objective of both functions. However, the parameter update will not be influenced by the NT-Xent loss if those respective gradients are (close to) zero. It is emphasized that this issue is important to investigate further to better understand if this is expected behavior in contrastive learned LSTM networks for de-novo molecular generation.

Further unpredicted behavior of the augmented SMILES strings was also found. After passing through the LSTM block of the network during forward propagation, the final hidden states of each respective pair of augments were extremely similar. Due to the complex computations performed by the LSTM, it is difficult to tell what the cause of such behavior is. If further work based on the implementation in this thesis is to be carried out, it is strongly suggested that this behaviour is properly investigated.

5.7 Future work

It is worth noting that the contrastive framework is not limited to the implementation of this thesis. Using the final hidden state from the LSTM layers can in many

ways be argued for is suitable when later applying the loss function, since it captures the entire SMILES sequence.

Other similar approaches that were considered but not tested due to time constraints were to use the hidden state at every time step t (output) of the final LSTM layer for contrastive learning. The output, in a similar sense to the final hidden state, contains information about the entire sequence and could therefore be a valid candidate in the use case of contrastive learning. It can be argued for that data representations throughout the network that contain some meaningful information about an input SMILES sequence is usable for maximizing the agreement, and by this principle it would also be feasible to use the output SMILES embeddings after the embedding layer.

A general approach that was not considered during the project was training in stages (pre-training). In this work, model training was performed in a single stage of forward propagation and BPTT under several epochs. A key issue that was faced during the training was that the NLL loss and the NT-Xent loss strongly affected each other. It could therefore be suitable to utilize contrastive learning as a form of pre-training for NLL loss to be applied in later steps. Pre-training approaches have shown to be proficient in settings with transformer architectures for de-novo molecule generational tasks. In practice, the pre-training could be applied to the entire network, or separate layers. However, each respective learning strategy minimizes loss in different aspects. It could therefore be suitable to pre-train specific layers, freeze them (learnable parameters are fixed and cannot be changed), and perform a final step of training. For example, pre-training and freezing embeddings under contrastive learning could perhaps maintain knowledge in the network of similar and dissimilar, and then finetune for the downstream task of de-novo molecular generation.

The findings of the experiment in which two methods of augmentation were used need to be further investigated. Although the results from the experiment hint that a combination of augmentations is superior to simply using a single method of augmentation for novelty, it cannot be concluded whether subgraphing or randomization on their own are better than the other. In addition to this, several other methods, such as edge perturbation, node drop, and attribute masking, should be tested to get a better grasp of the limitations of contrastive learning in this setting. Although testing a large number of combinations requires a large amount of computing power, exploring the limits for molecular de-novo generative models should remain a high priority.

6

Conclusion

This thesis leveraged AstraZeneca's current LSTM neural network to incorporate representation learning in the form of contrastive learning. This required building algorithms to draw augmentations of SMILES data to obtain positive, low contrast pairs. In accordance to this, the core algorithm to compute the NT-Xent loss that strives to pull together positive pairs and repel negative pairs was created. In addition to this, the network was modified by creating a custom forward propagation function to process SMILES augmentations to obtain their final hidden states, and further reduced the dimension (and noise) of said state by adding a second fully connected layer to the network. To assess the performance of the models, benchmarks were carried out through a self-implemented benchmarking suite for validity, uniqueness, novelty, internal diversity and FCD were implemented. The MOSES benchmarking suite was also used in accordance with that dataset of SMILES. This methodology was conducted to answer the following research questions:

- **RQ1:** Does a model trained to minimize the negative log likelihood of SMILES strings **and** trained to maximize agreement between positive SMILES representations on a LSTM neural network prove superior to models trained only to minimize the negative log likelihood on LSTM neural networks for SMILES string generation in the REINVENT platform, in generating SMILES with greater novelty, diversity and validity?

- **RQ2:** Does training to maximize agreement between SMILES representations for the SMILES generation task benefit from using more than one method of SMILES augmentation by using different, and more positive pairs of SMILES representation during training, in regards to novelty, diversity and validity?

It can therefore by the work of this thesis be concluded that:

- **RQ1:** Incorporating contrastive learning into the current REINVENT LSTM neural network did not prove to be superior nor inferior in terms of novelty, diversity and validity in comparison to the REINVENT trained models. Due to high scores on uniqueness, it cannot be properly deduced if contrastive learning has produced fewer enumerations of SMILES.

- **RQ2:** Using a combination of randomization and subgraphing gives hints

that using more than one augmentation technique can prove superior to using a single augmentation method, and could prove superior to core LSTM neural networks for the de-novo molecular generation task, but does not explicitly reveal their stand-alone superiority.

The purpose of this thesis was to address if representation learning through contrastive learning is applicable on LSTM neural networks with the downstream task of de-novo molecular generation, and to further understand its capabilities and limitations in this setting. We believe that this thesis has contributed with hints that contrastive learning is applicable in LSTM neural networks to build stronger, and more generalized foundation models in terms of novelty and drug-likeliness. Due to scores on uniqueness being either very similar between learning strategies or close to 100% across experiments, it cannot be determined if contrastive learning has decreased the amount of SMILES enumerations in the generated sets. This work has laid ground for future appliances of contrastive learning in the endeavour of finding future drug candidates in LSTM neural networks.

Given the reliability of the conducted experiments, it is crucial that future work consider broadening the testing space, with other suggested methods of training, and possibly other methods of augmentations. On top of this, it is recommended that models based on the projects implementation are thoroughly assessed its further use case in narrowed drug space searches before industry deployments.

Bibliography

[1] M. Olivecrona, T. Blaschke, O. Engkvist, and H. Chen, "Molecular de-novo design through deep reinforcement learning," *Journal of cheminformatics*, vol. 9, pp. 1–14, 2017.

[2] T. Blaschke, J. Arús-Pous, H. Chen, *et al.*, "Reinvent 2.0: An ai tool for de novo drug design," *Journal of chemical information and modeling*, vol. 60, no. 12, pp. 5918–5922, 2020.

[3] H. H. Loeffler, J. He, A. Tibo, *et al.*, "Reinvent 4: Modern ai–driven generative molecule design," *Journal of Cheminformatics*, vol. 16, no. 1, p. 20, 2024.

[4] Y. Qian, M. Shi, and Q. Zhang, "Consmi: Contrastive learning in the simplified molecular input line entry system helps generate better molecules," *Molecules*, vol. 29, no. 2, 2024, ISSN: 1420-3049. DOI: 10.3390/molecules29020495. [Online]. Available: https://www.mdpi.com/1420-3049/29/2/495.

[5] R. J. Williams and D. Zipser, "A learning algorithm for continually running fully recurrent neural networks," *Neural computation*, vol. 1, no. 2, pp. 270–280, 1989.

[6] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, "A simple framework for contrastive learning of visual representations," in *International conference on machine learning*, PMLR, 2020, pp. 1597–1607.

[7] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick, "Momentum contrast for unsupervised visual representation learning," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 9729–9738.

[8] J.-B. Grill, F. Strub, F. Altché, *et al.*, "Bootstrap your own latent-a new approach to self-supervised learning," *Advances in neural information processing systems*, vol. 33, pp. 21 271–21 284, 2020.

[9] D. Weininger, "Smiles, a chemical language and information system. 1. introduction to methodology and encoding rules," *Journal of chemical information and computer sciences*, vol. 28, no. 1, pp. 31–36, 1988.

[10] D. Weininger, A. Weininger, and J. L. Weininger, "Smiles. 2. algorithm for generation of unique smiles notation," *Journal of chemical information and computer sciences*, vol. 29, no. 2, pp. 97–101, 1989.

[11] J. Bajorath, *Chemoinformatics: concepts, methods, and tools for drug discovery*. Springer Science & Business Media, 2008, vol. 275.

[12] F.-F. Li, J. Johnson, and S. Yeung, *Lecture 10*, Stanford University, available at: https://cs231n.stanford.edu/slides/2017/cs231n$_2$017$_l$ecture10.*pdf*, May 2017.

[13] O. Vinyals, A. Toshev, S. Bengio, and D. Erhan, "Show and tell: A neural image caption generator," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 3156–3164.

[14] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," *Advances in neural information processing systems*, vol. 27, 2014.

[15] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016.

[16] K. Greff, R. K. Srivastava, J. Koutník, B. R. Steunebrink, and J. Schmidhuber, "Lstm: A search space odyssey," *IEEE transactions on neural networks and learning systems*, vol. 28, no. 10, pp. 2222–2232, 2016.

[17] C. Shorten and T. M. Khoshgoftaar, "A survey on image data augmentation for deep learning," *Journal of Big Data*, vol. 6, no. 1, p. 60, 2019.

[18] E. J. Bjerrum, "Smiles enumeration as data augmentation for neural network modeling of molecules," *arXiv preprint arXiv:1703.07076*, 2017.

[19] Y. You, T. Chen, Y. Sui, T. Chen, Z. Wang, and Y. Shen, "Graph contrastive learning with augmentations," in *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, Eds., vol. 33, Curran Associates, Inc., 2020, pp. 5812–5823. [Online]. Available: `https://proceedings.neurips.cc/paper_files/paper/2020/file/3fe230348e9a12c13120749e3f9fa4cd-Paper.pdf`.

[20] T. Gao, X. Yao, and D. Chen, "Simcse: Simple contrastive learning of sentence embeddings," *arXiv preprint arXiv:2104.08821*, 2021.

[21] C. M. Bishop, *Pattern Recognition and Machine Learning*. Springer, 2006.

[22] T. St-Hilaire, *Simple self-supervised learning*, `https://sthalles.github.io/simple-self-supervised-learning/`, Accessed: 2025-06-06, 2020.

[23] H. Hotelling, "Analysis of a complex of statistical variables into principal components," *Journal of Educational Psychology*, vol. 24, no. 6, pp. 417–441, 1933.

[24] K. Pearson, "On lines and planes of closest fit to systems of points in space," *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, vol. 2, no. 11, pp. 559–572, 1901.

[25] M. Wattenberg, F. Viégas, and I. Johnson. "How to use t-sne effectively." Accessed: March 2025. (2016), [Online]. Available: `https://distill.pub/2016/misread-tsne/`.

[26] L. McInnes, J. Healy, and J. Melville, "Umap: Uniform manifold approximation and projection for dimension reduction," *arXiv preprint arXiv:1802.03426*, 2018.

[27] BenevolentAI, "Guacamol: A new framework to compare models for de novo molecular design," *BenevolentAI Blog*, 2019. [Online]. Available: `https://www.benevolent.com/news-and-media/blog-and-videos/guacamol-new-framework-compare-models-de-novo-molecular-design/`.

[28] D. Polykovskiy, A. Zhebrak, B. Sanchez-Lengeling, *et al.*, "Molecular sets (moses): A benchmarking platform for molecular generation models," *Frontiers in pharmacology*, vol. 11, p. 565 644, 2020.

[29] K. Preuer, P. Renz, T. Unterthiner, S. Hochreiter, and G. Klambauer, "Fréchet chemnet distance: A metric for generative models for molecules in drug discov-

ery," *Journal of chemical information and modeling*, vol. 58, no. 9, pp. 1736–1741, 2018.

[30] Y. Wang, S. H. Bryant, T. Cheng, *et al.*, "Pubchem bioassay: 2017 update," *Nucleic acids research*, vol. 45, no. D1, pp. D955–D963, 2017.

[31] J. J. Irwin, T. Sterling, M. M. Mysinger, E. S. Bolstad, and R. G. Coleman, "Zinc: A free tool to discover chemistry for biology," *Journal of chemical information and modeling*, vol. 52, no. 7, pp. 1757–1768, 2012.

[32] B. Zdrazil, E. Felix, F. Hunter, *et al.*, "The chembl database in 2023: A drug discovery platform spanning multiple bioactivity data types and time periods," *Nucleic Acids Research*, vol. 52, no. D1, pp. D1180–D1192, Nov. 2023, ISSN: 0305-1048. DOI: 10.1093/nar/gkad1004. eprint: https://academic.oup.com/nar/article-pdf/52/D1/D1180/55040046/gkad1004.pdf. [Online]. Available: https://doi.org/10.1093/nar/gkad1004.

[33] S. K. Bhal, "Logpmaking sense of the value," Advanced Chemistry Development, Inc. (ACD/Labs), Toronto, ON, Canada, Tech. Rep., 2019, Application Note. [Online]. Available: https://www.acdlabs.com/wp-content/uploads/download/app/physchem/making_sense.pdf.

[34] R. Wang, *Xlogp v2.1 introduction*, https://ics.uci.edu/~dock/manuals/xlogp2.1/intro.html, Accessed: 2025-05-05, 2001.

[35] J. Clayden, N. Greeves, and S. Warren, *Organic chemistry*. Oxford university press, 2012.

[36] ZINC, *Clean leads*, Accessed: 28-Feb-2025, n.d. [Online]. Available: https://zinc12.docking.org/subsets/clean-leads.

[37] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[38] G. Landrum, *Rdkit: Open-source cheminformatics*, https://www.rdkit.org, Accessed: 2025-05-17, 2006–.

[39] D. Polykovskiy, A. Zhebrak, B. Sanchez-Lengeling, *et al.*, *MOSES: A Benchmarking Platform for Molecular Generation Models*, https://github.com/molecularsets/moses, Accessed: 2025-06-06, 2020.

[40] P. Foundation, *Torch.nn.nllloss*, Accessed: March 17, 2025, 2024. [Online]. Available: https://pytorch.org/docs/stable/generated/torch.nn.NLLLoss.html.

[41] Optuna, *Optuna: A hyperparameter optimization framework*, Accessed: 2025-03-26, 2019. [Online]. Available: https://optuna.org/.

A

Physicochemical properties from generated SMILES strings

A.1 Physicochemical Properties: Baseline dataset

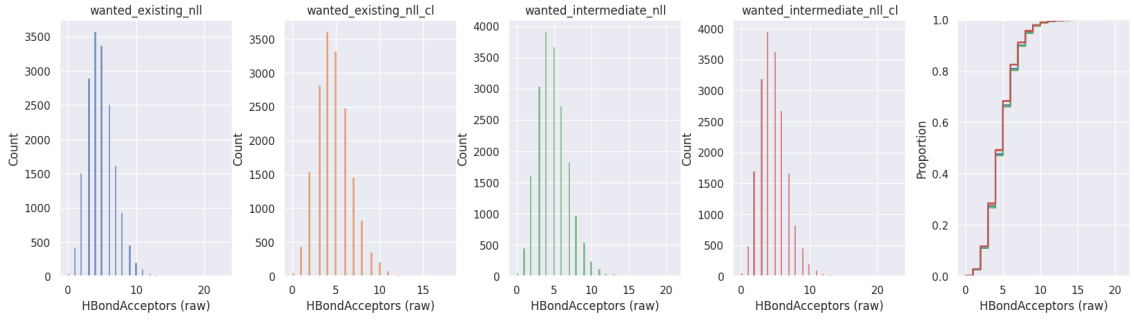


Figure A.1: Hydrogen Bond Acceptors

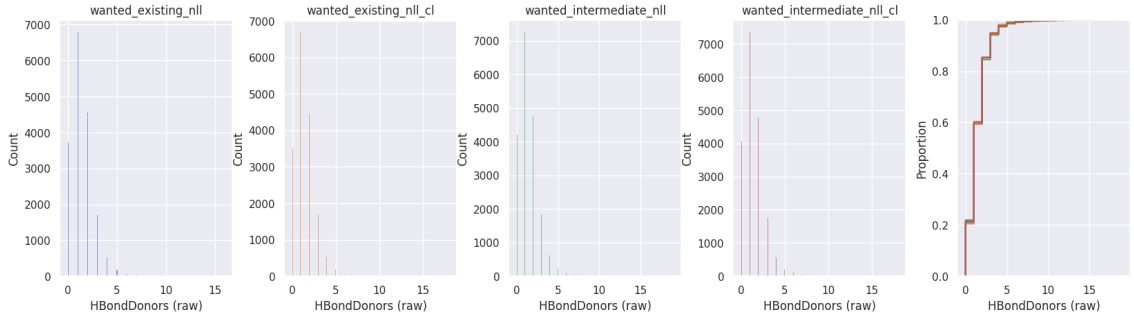


Figure A.2: Hydrogen Bond Donors

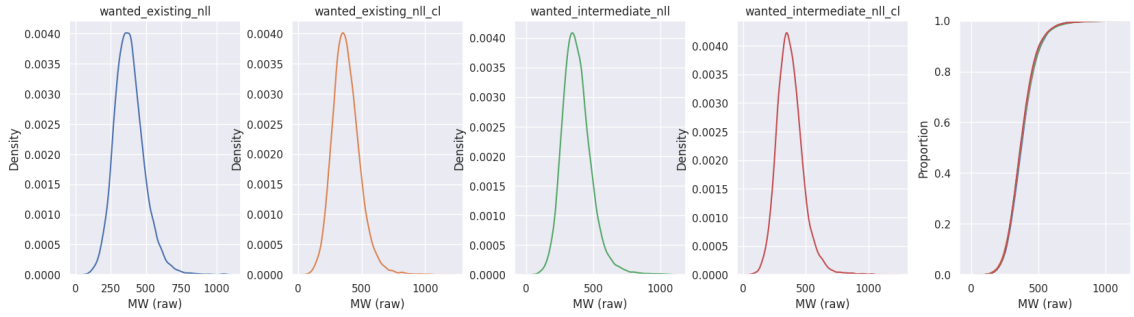


Figure A.3: Molecular Weight

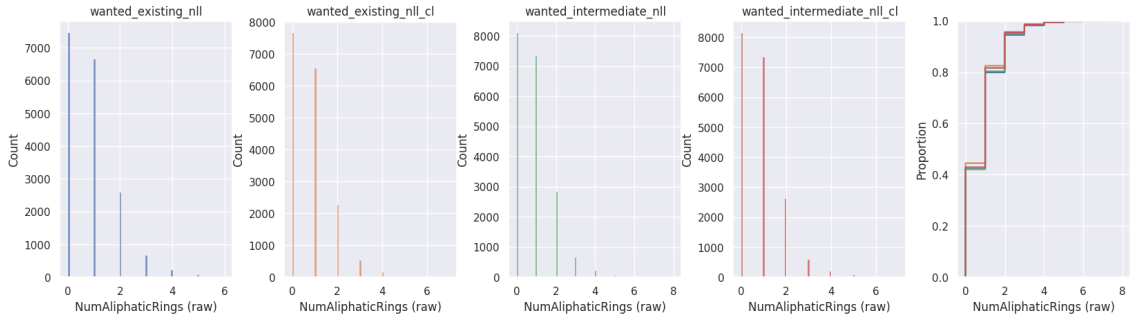


Figure A.4: Number of Aliphatic Rings

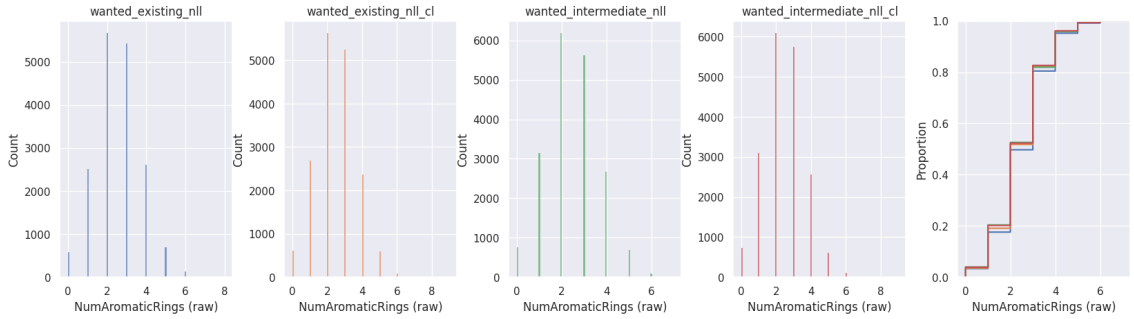


Figure A.5: Number of Aromatic Rings

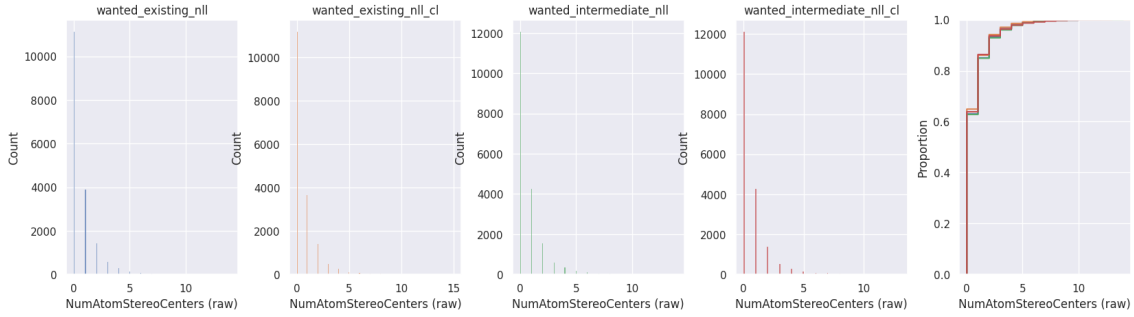


Figure A.6: Number of Atom Stereocenters

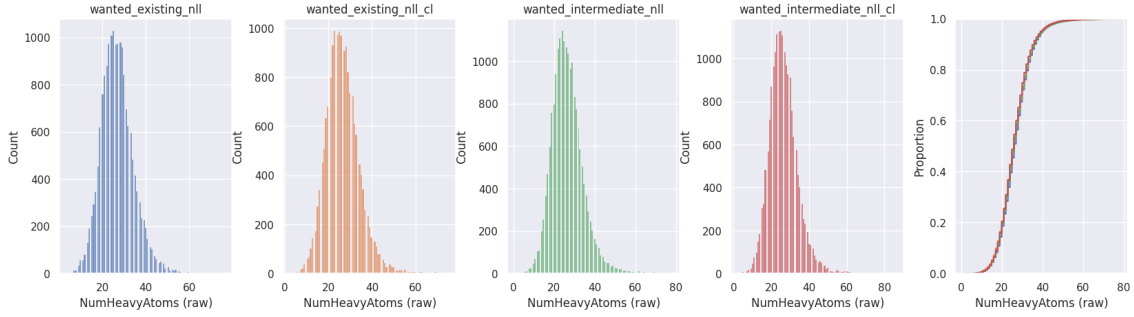


Figure A.7: Number of Heavy Atoms

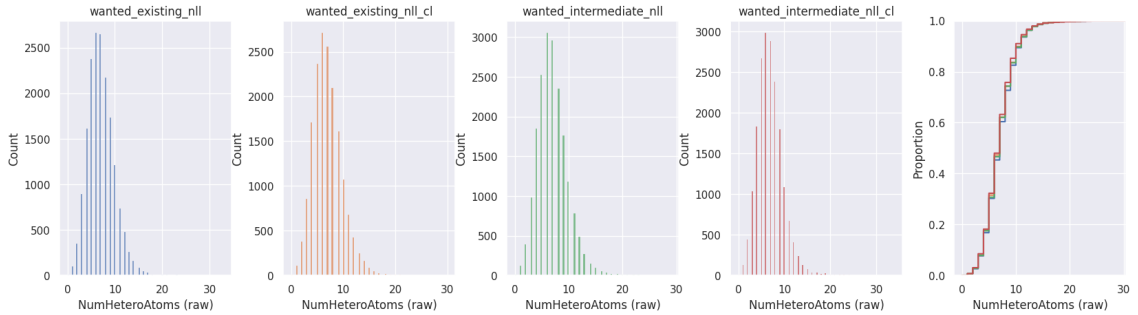


Figure A.8: Number of Hetero Atoms

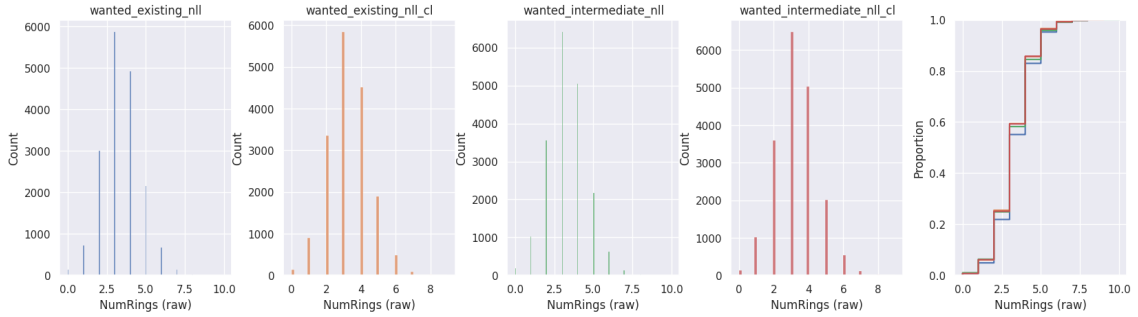


Figure A.9: Number of Rings

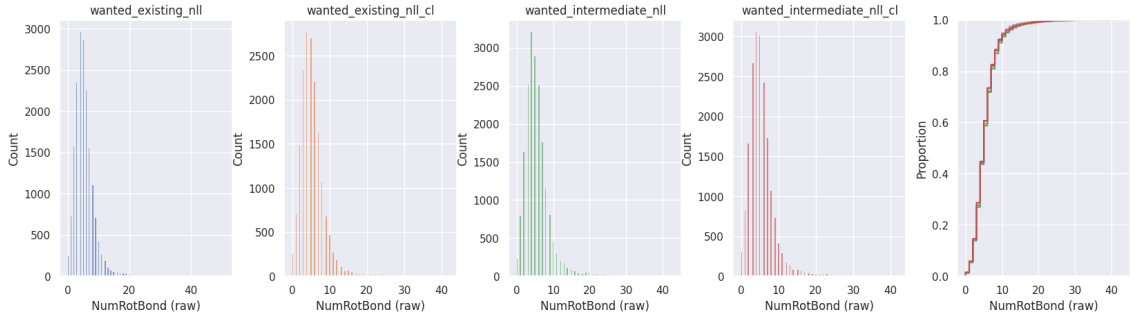


Figure A.10: Number of Rotatable Bonds

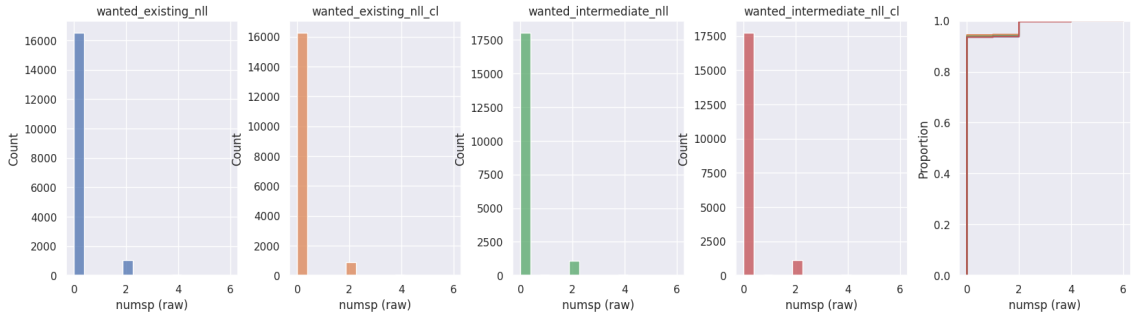


Figure A.11: sp Hybridization

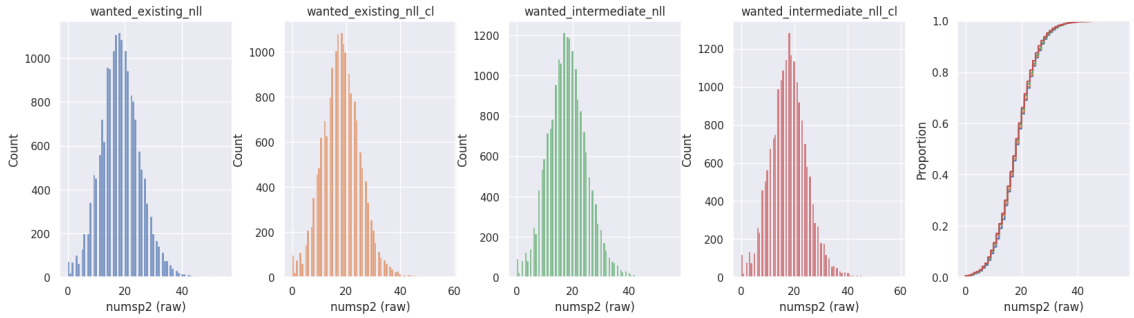


Figure A.12: sp2 Hybridization

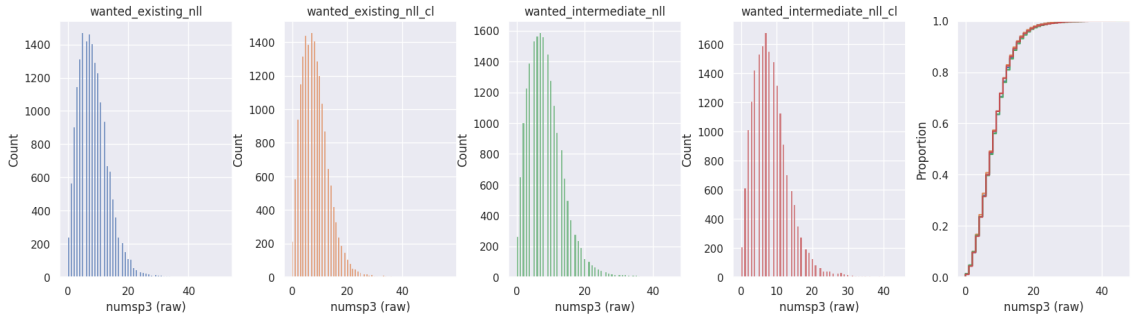


Figure A.13: sp3 Hybridization

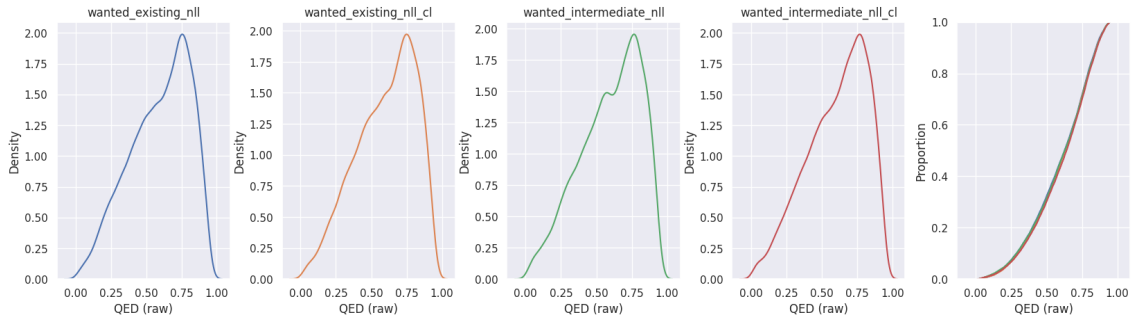


Figure A.14: Quantitative Estimate of Drug-likeness (QED)

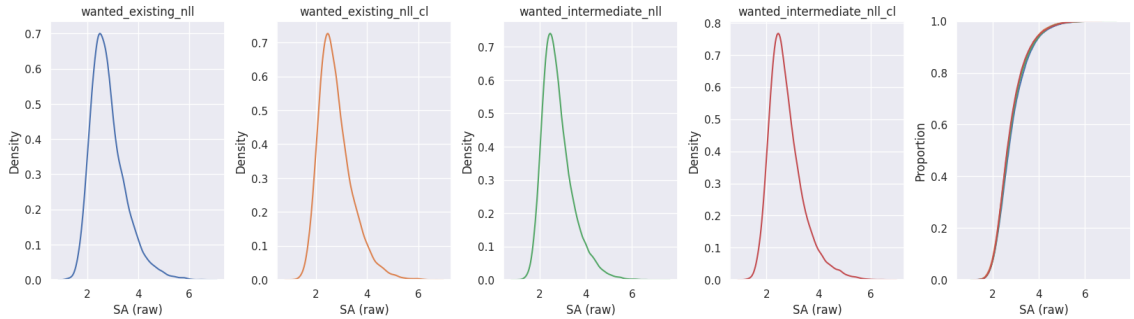


Figure A.15: Synthetic Accessibility

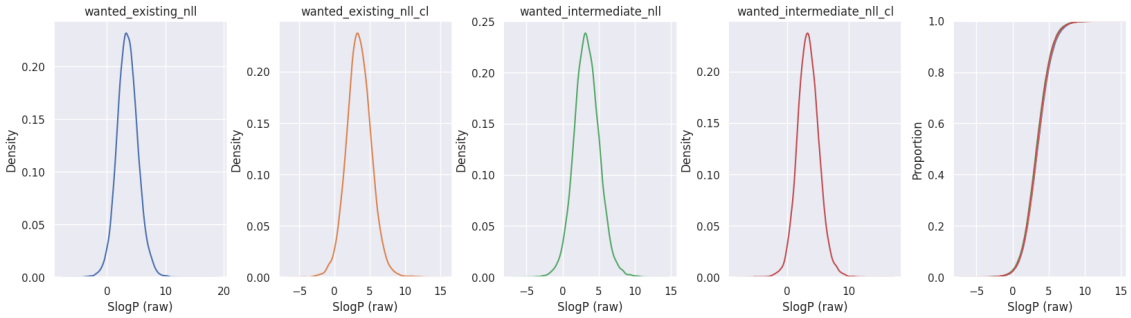


Figure A.16: SlogP (Octanol-Water Partition Coefficient)

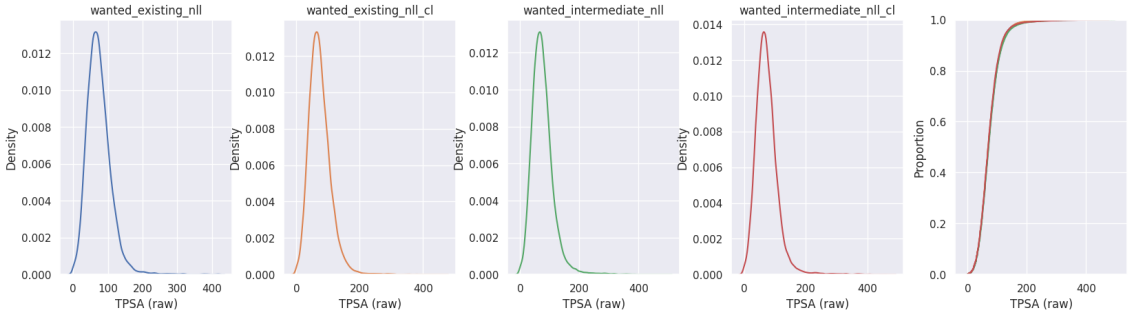


Figure A.17: Topological Polar Surface Area (TPSA)

A.2 Physicochemical Properties: MOSES dataset

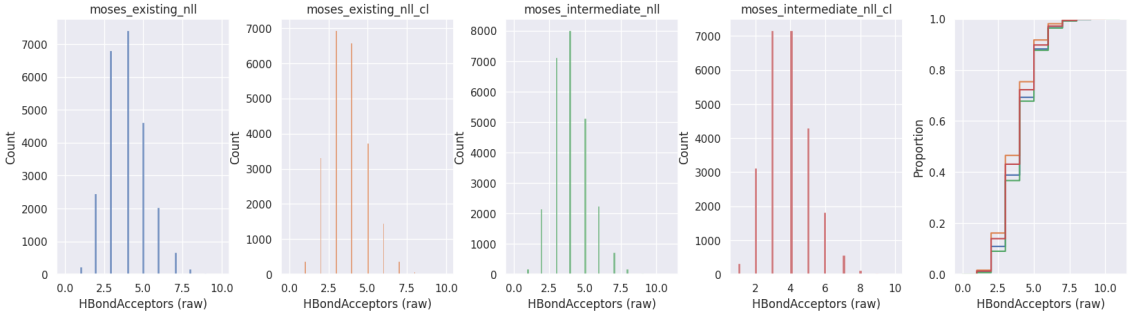


Figure A.18: Hydrogen Bond Acceptors

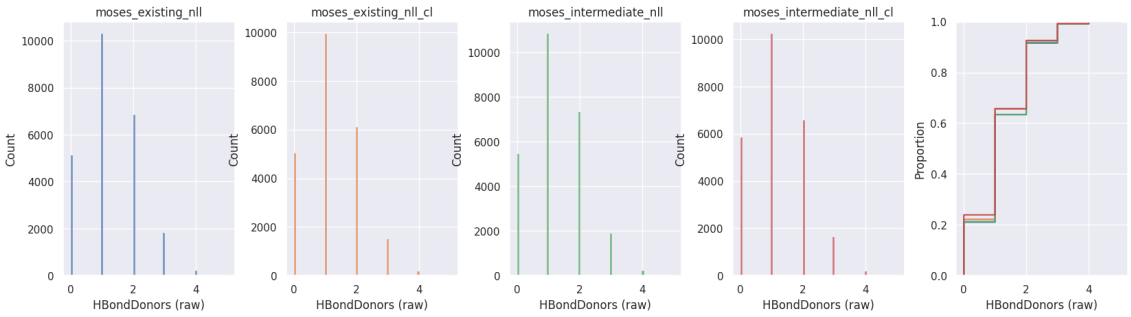


Figure A.19: Hydrogen Bond Donors

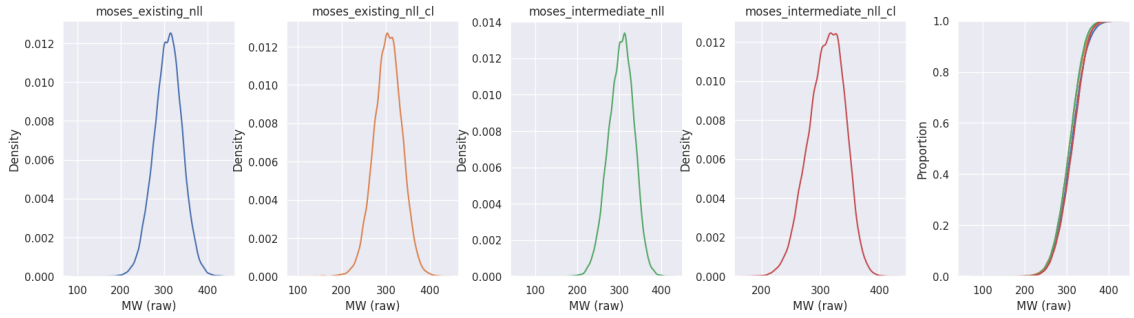


Figure A.20: Molecular Weight

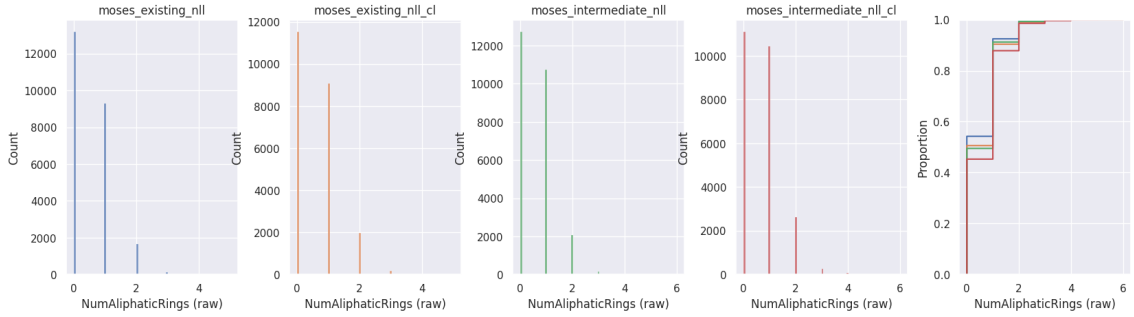


Figure A.21: Number of Aliphatic Rings

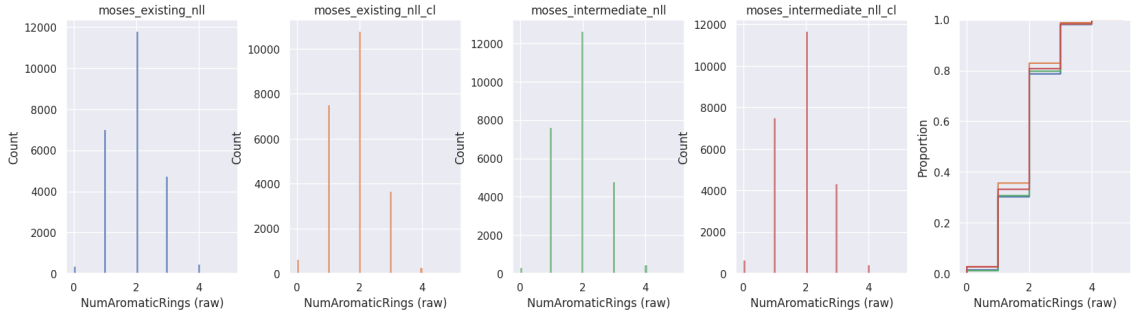


Figure A.22: Number of Aromatic Rings

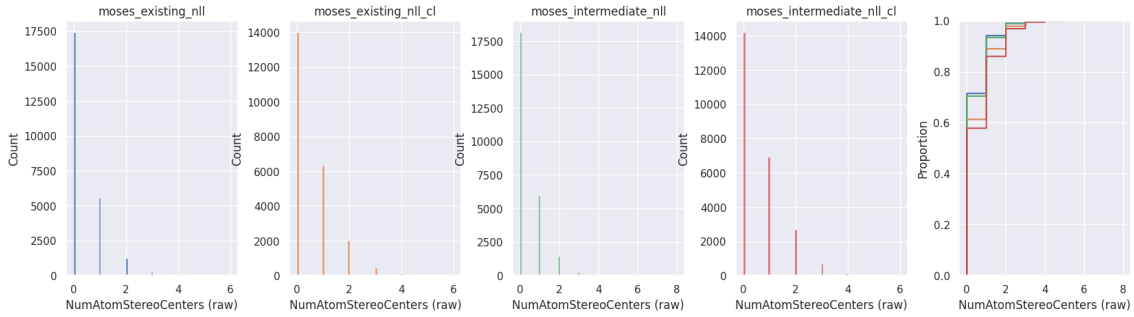


Figure A.23: Number of Atom Stereocenters

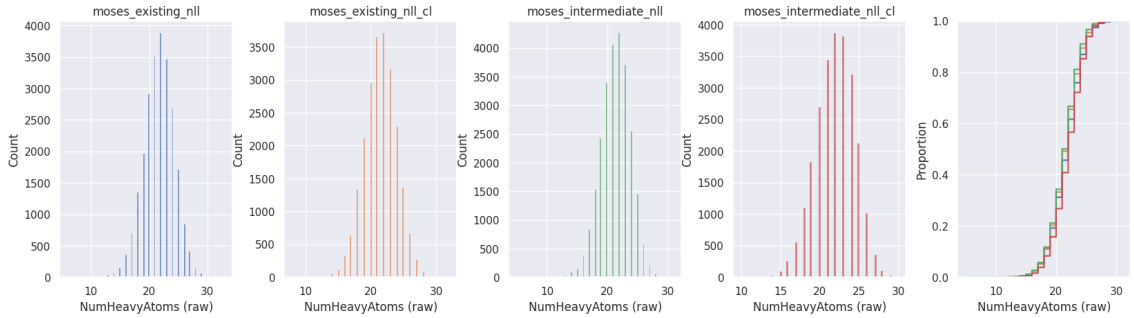


Figure A.24: Number of Heavy Atoms

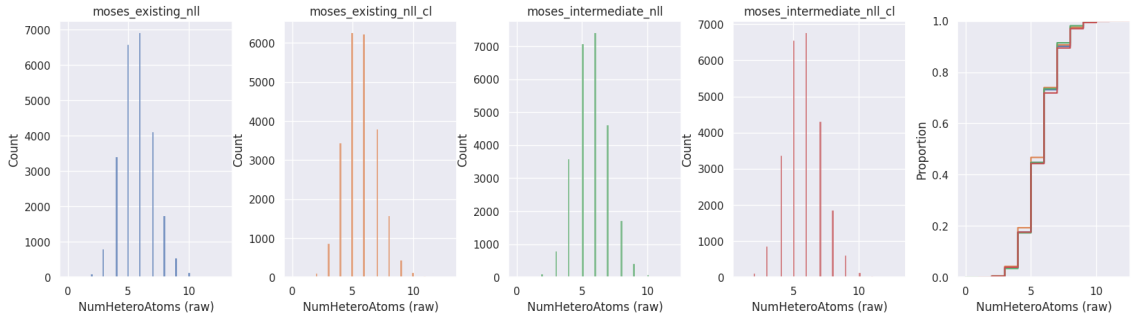


Figure A.25: Number of Hetero Atoms

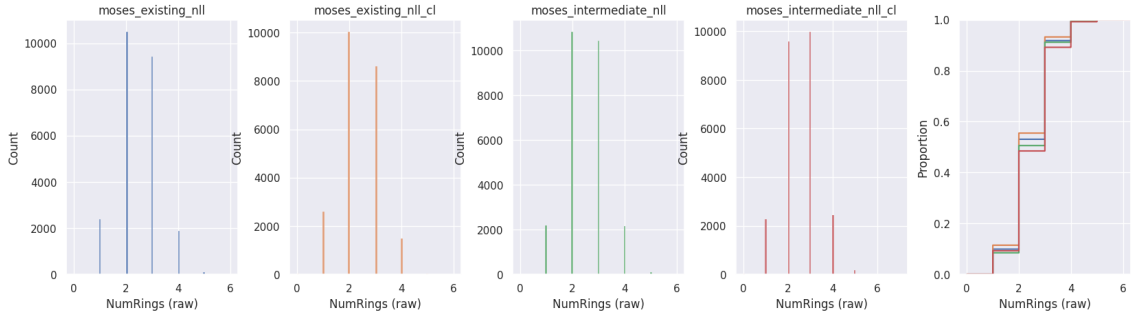


Figure A.26: Number of Rings

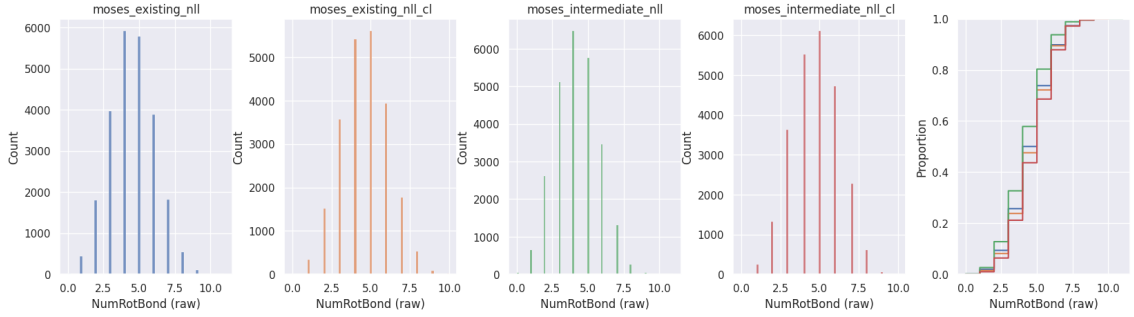


Figure A.27: Number of Rotatable Bonds

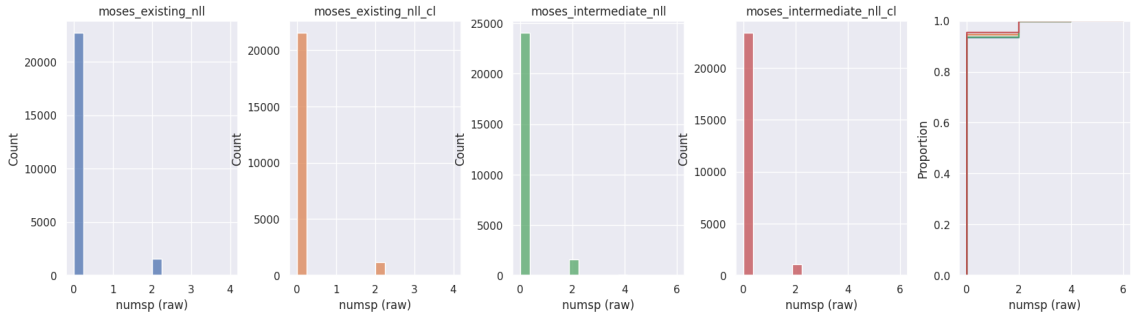


Figure A.28: sp Hybridization

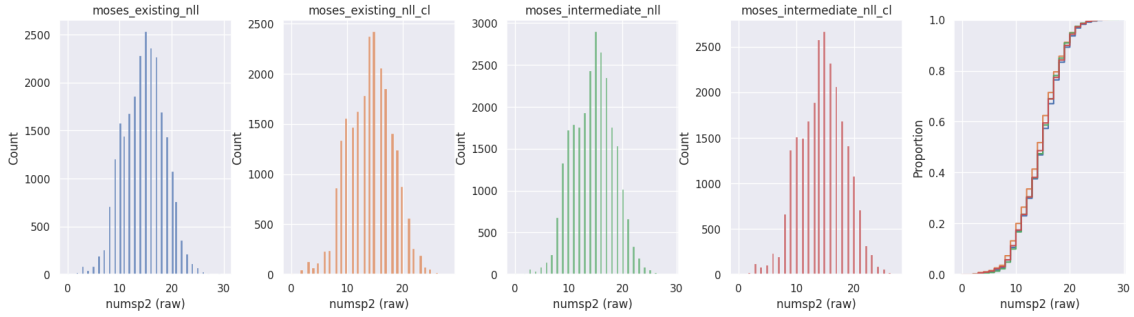


Figure A.29: sp2 Hybridization

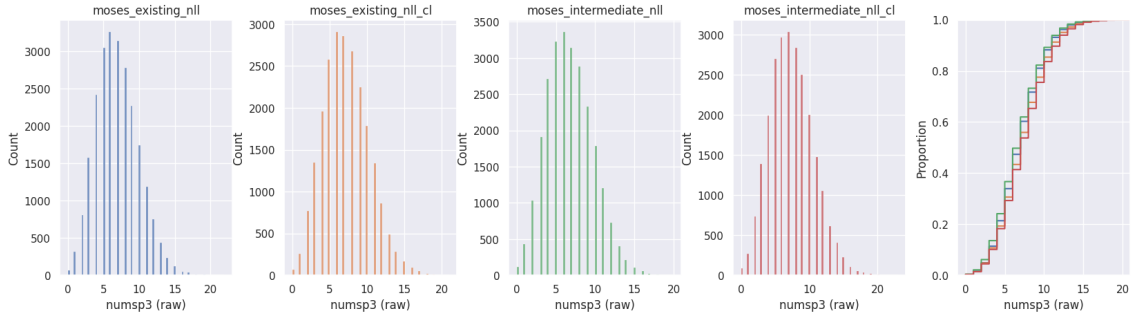


Figure A.30: sp3 Hybridization

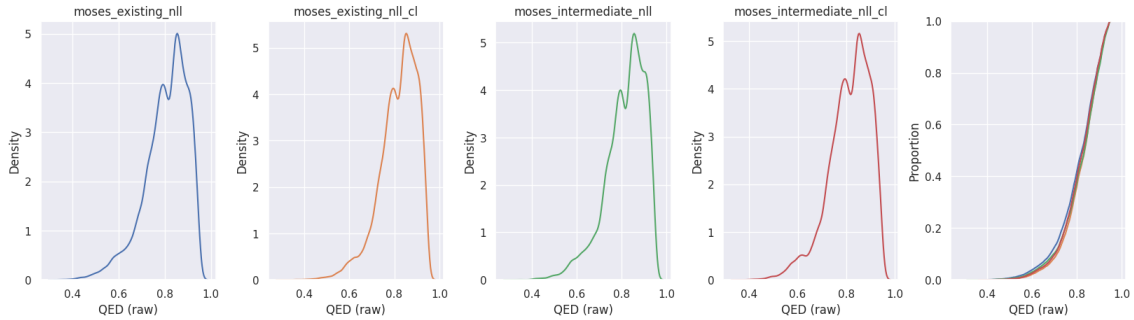


Figure A.31: Quantitative Estimate of Drug-likeness (QED)

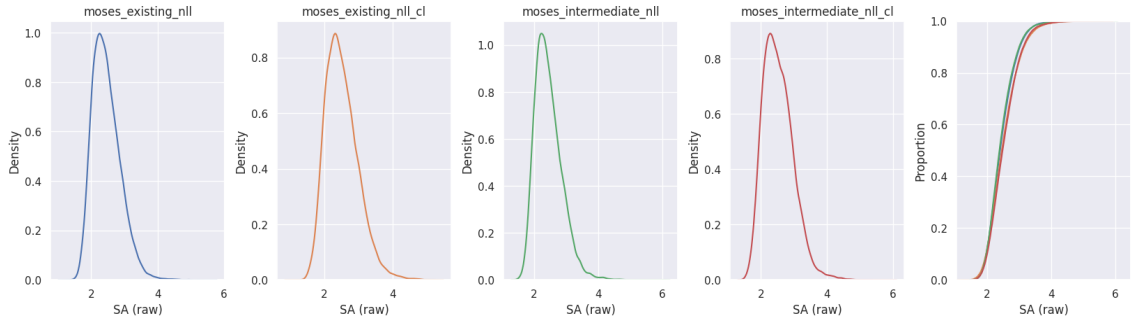


Figure A.32: Synthetic Accessibility

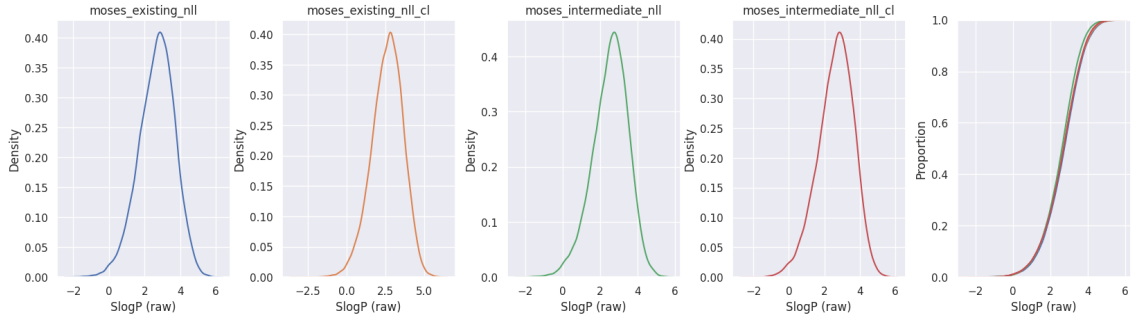


Figure A.33: SlogP (Octanol-Water Partition Coefficient)

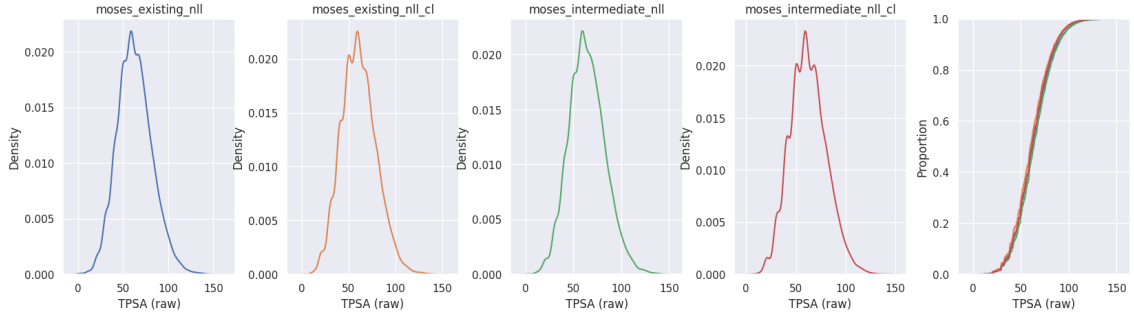


Figure A.34: Topological Polar Surface Area (TPSA)

A.3 Physicochemical Properties: GuacaMol dataset

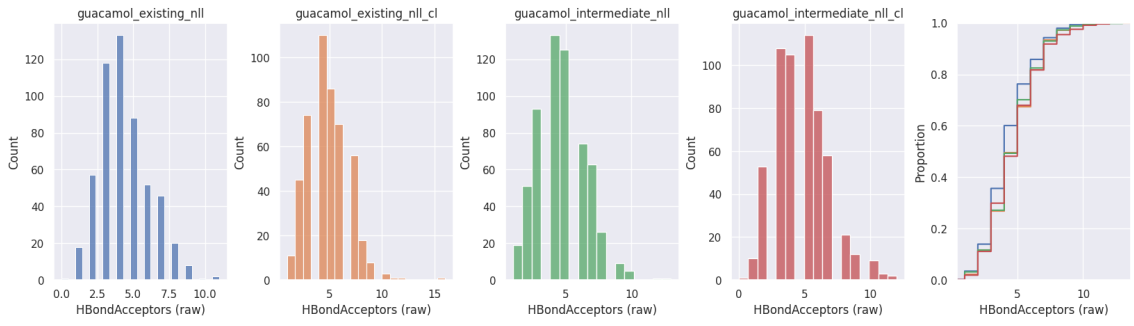


Figure A.35: Hydrogen Bond Acceptors

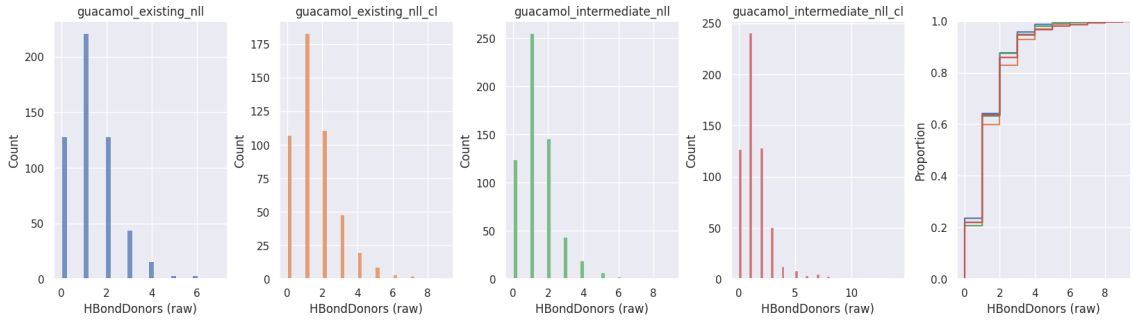


Figure A.36: Hydrogen Bond Donors

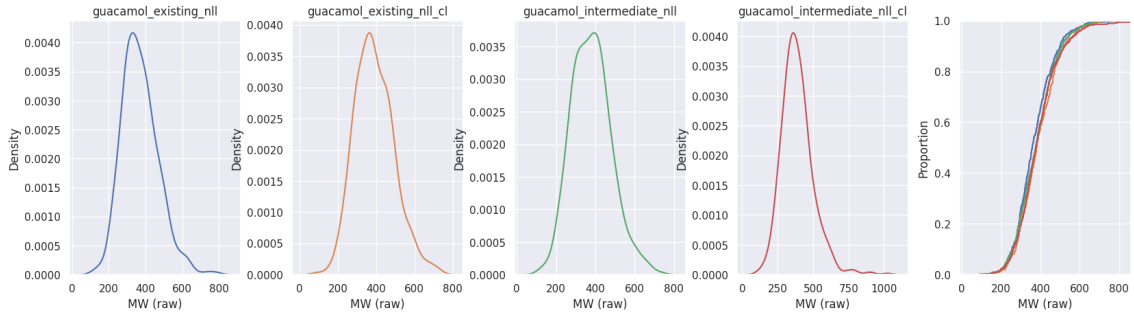


Figure A.37: Molecular Weight

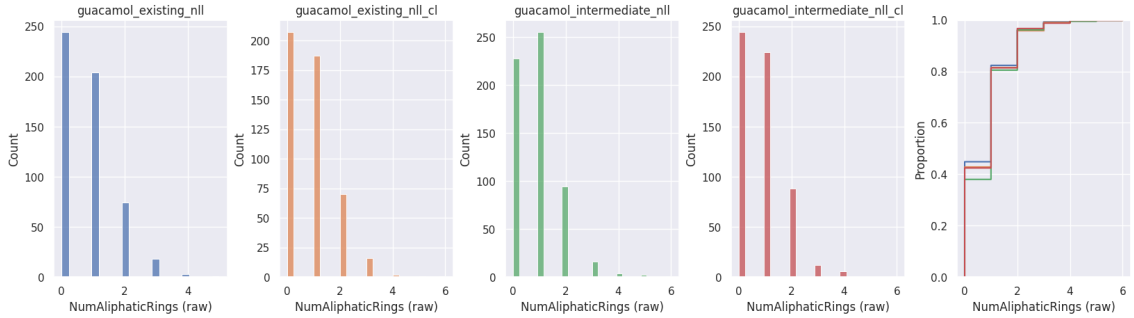


Figure A.38: Number of Aliphatic Rings

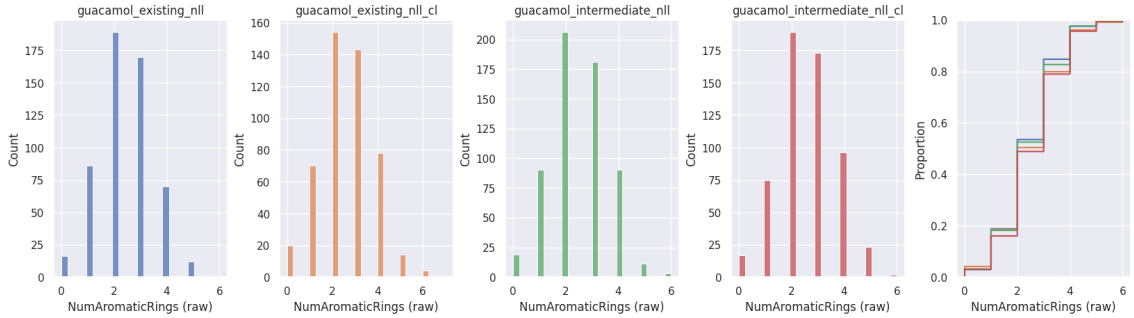


Figure A.39: Number of Aromatic Rings

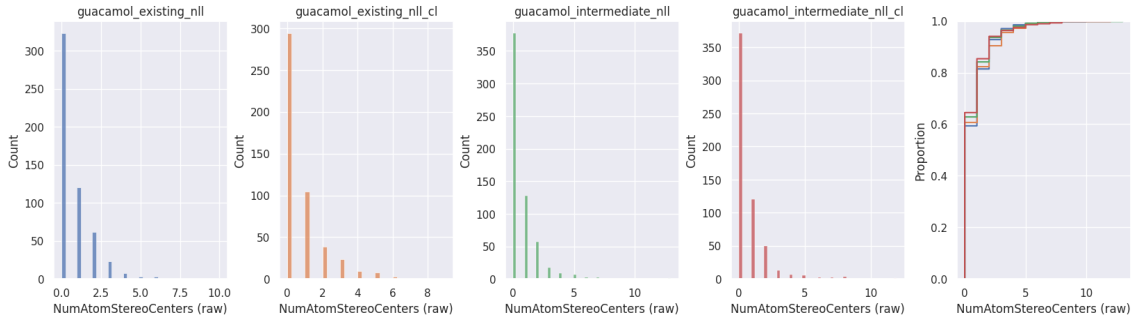


Figure A.40: Number of Atom Stereocenters

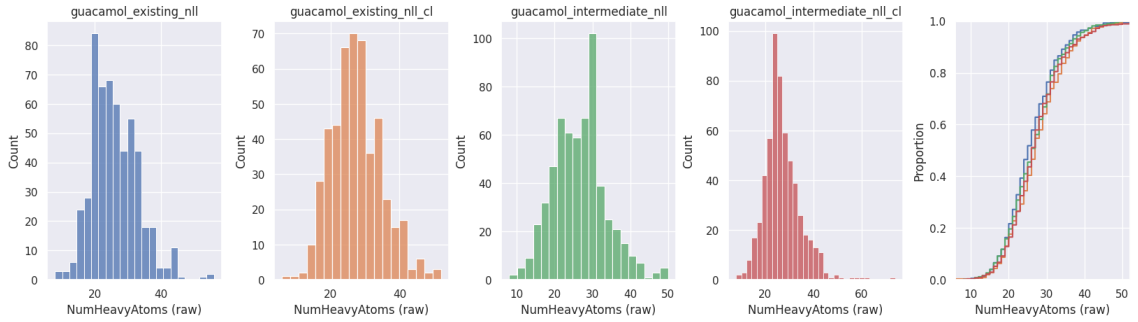


Figure A.41: Number of Heavy Atoms

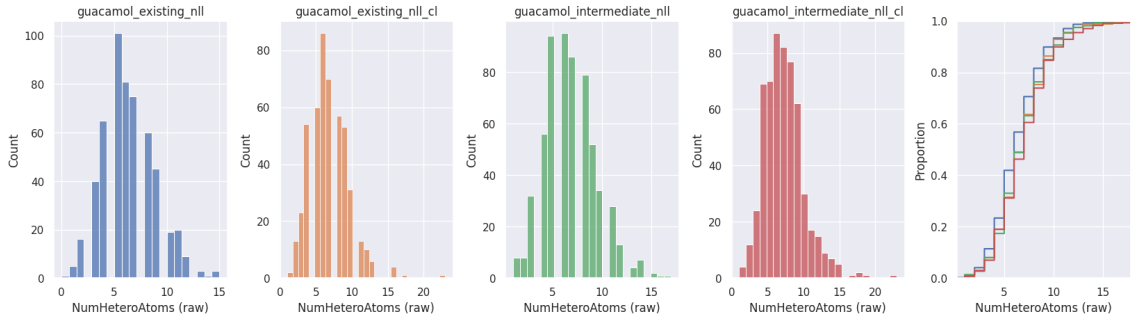


Figure A.42: Number of Hetero Atoms

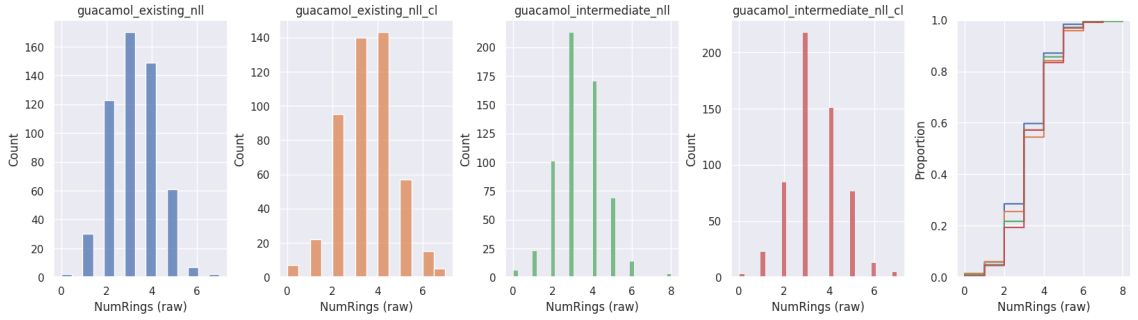


Figure A.43: Number of Rings

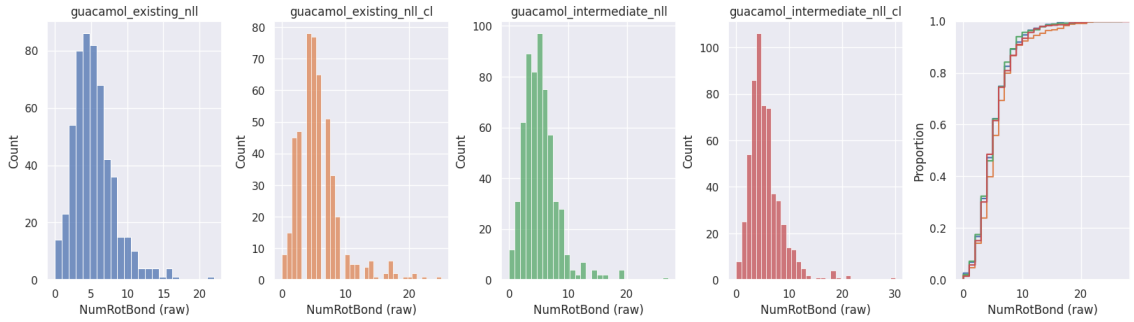


Figure A.44: Number of Rotatable Bonds

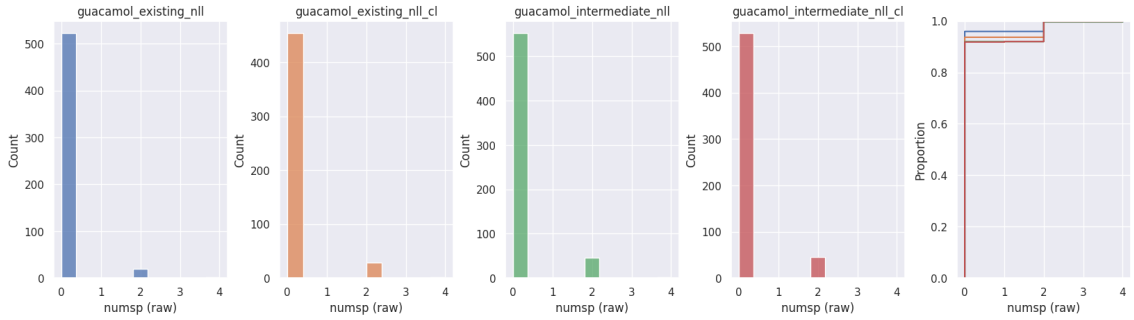


Figure A.45: sp Hybridization

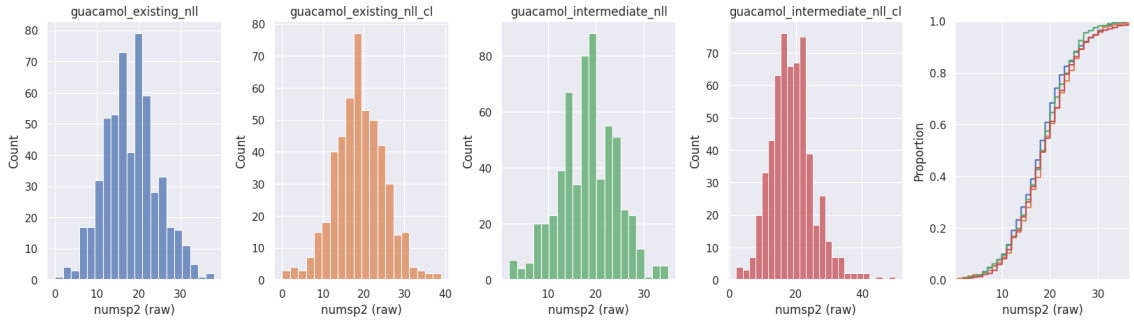


Figure A.46: sp2 Hybridization

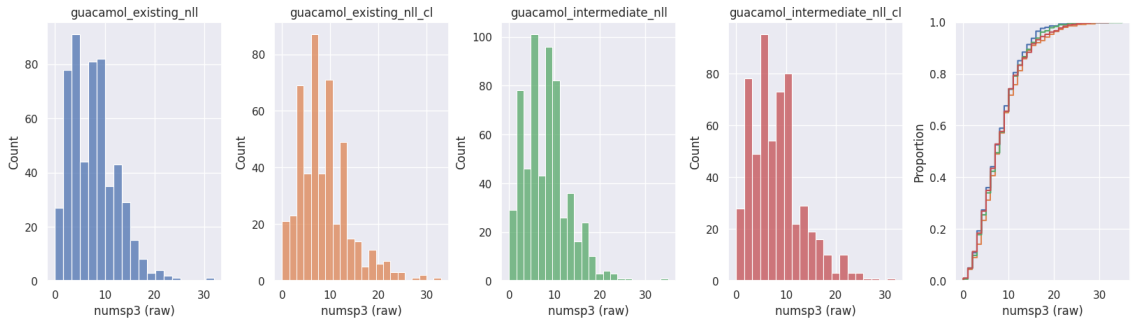


Figure A.47: sp3 Hybridization

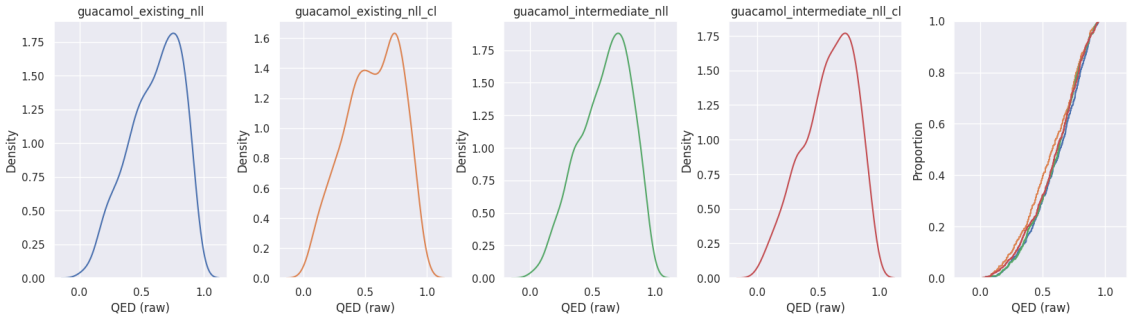


Figure A.48: Quantitative Estimate of Drug-likeness (QED)

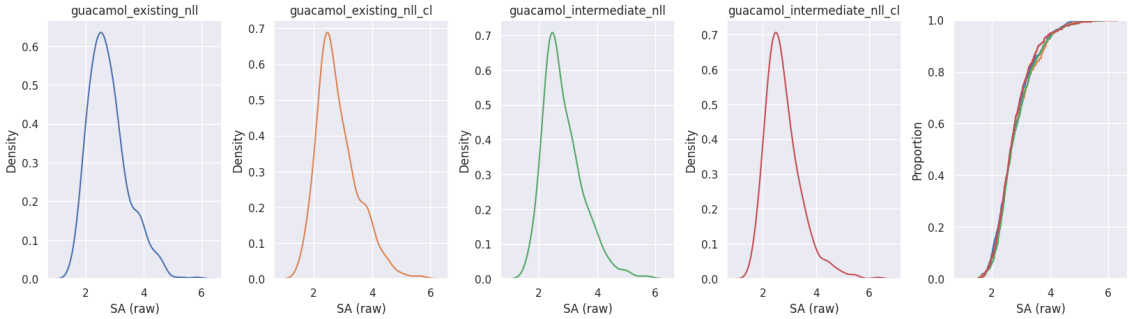


Figure A.49: Synthetic Accessibility

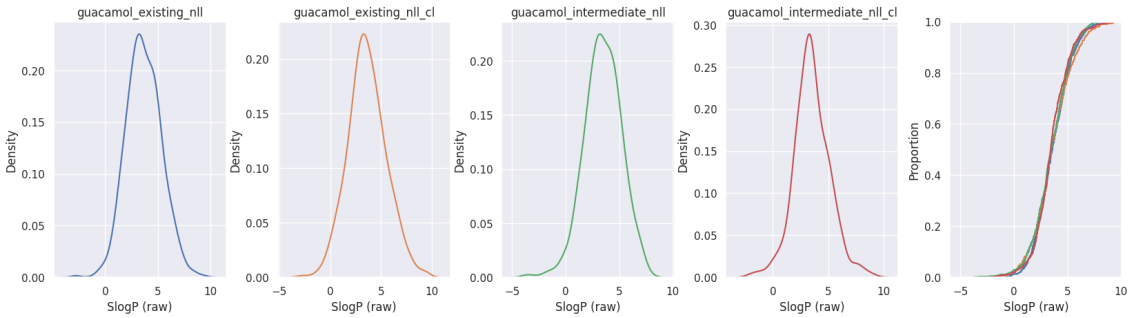


Figure A.50: SlogP (Octanol-Water Partition Coefficient)

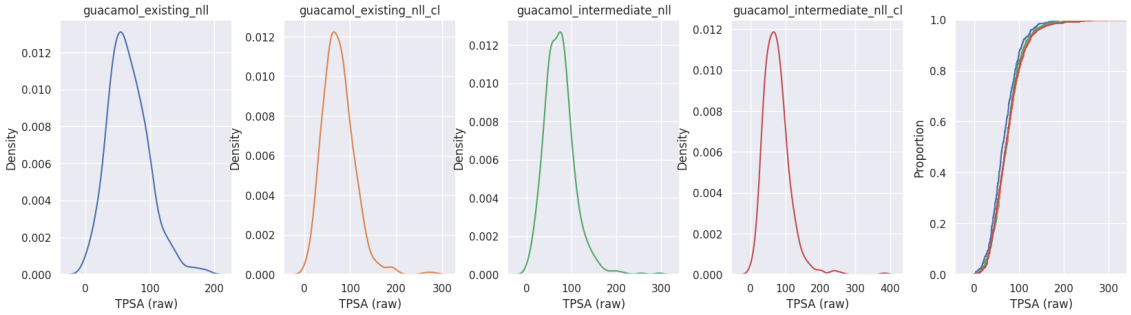


Figure A.51: Topological Polar Surface Area (TPSA)

A.4 Physicochemical Properties: Baseline dataset with several augmentations

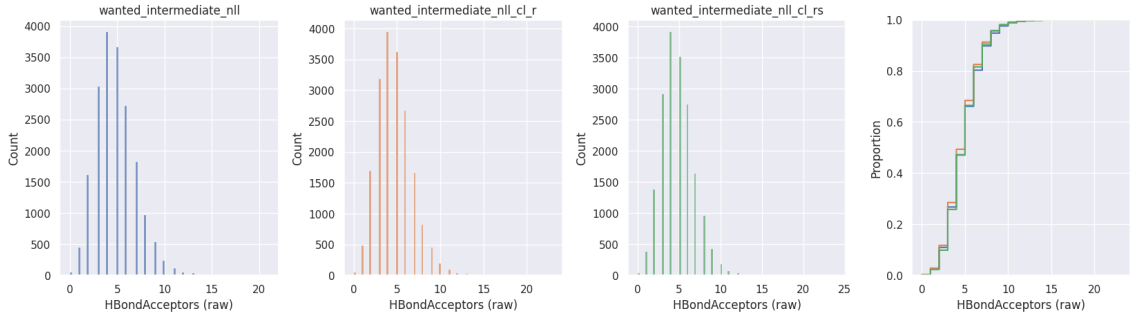


Figure A.52: Hydrogen Bond Acceptors

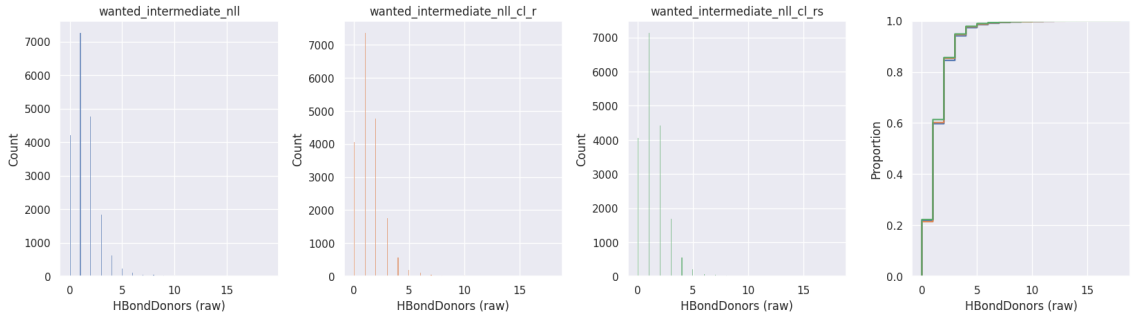


Figure A.53: Hydrogen Bond Donors

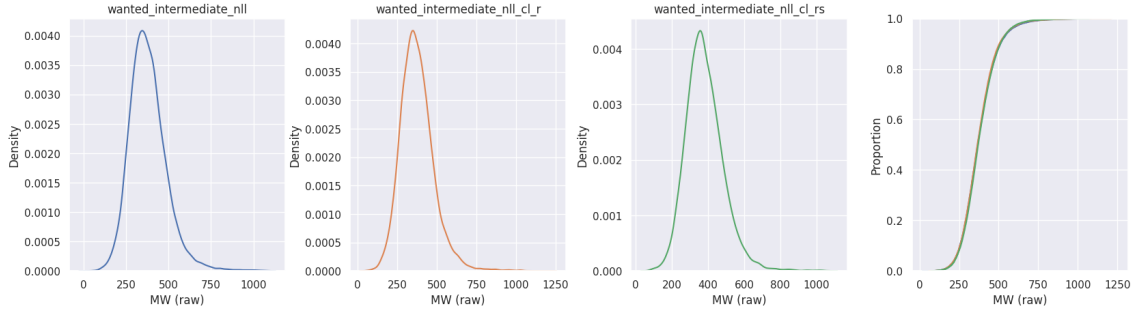


Figure A.54: Molecular Weight

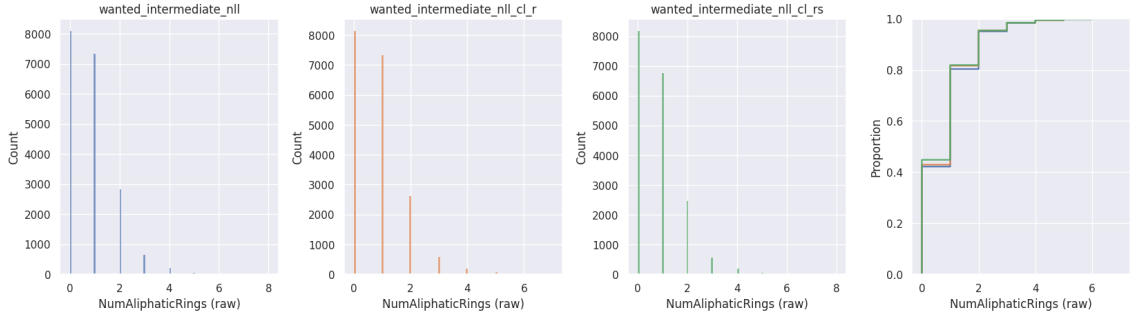


Figure A.55: Number of Aliphatic Rings

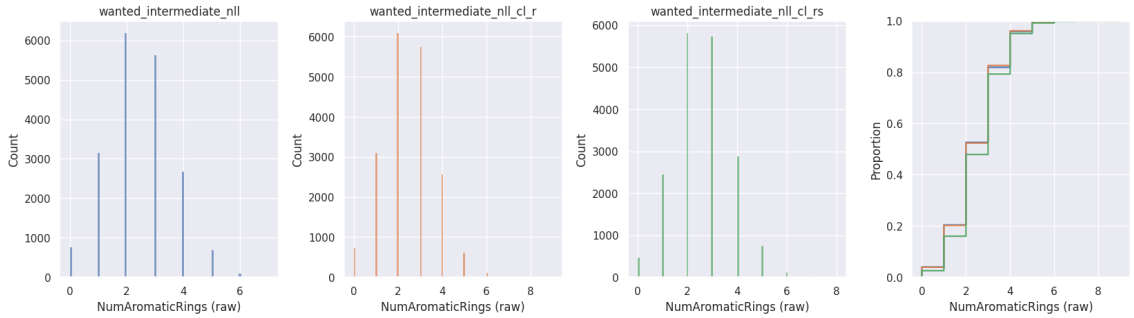


Figure A.56: Number of Aromatic Rings

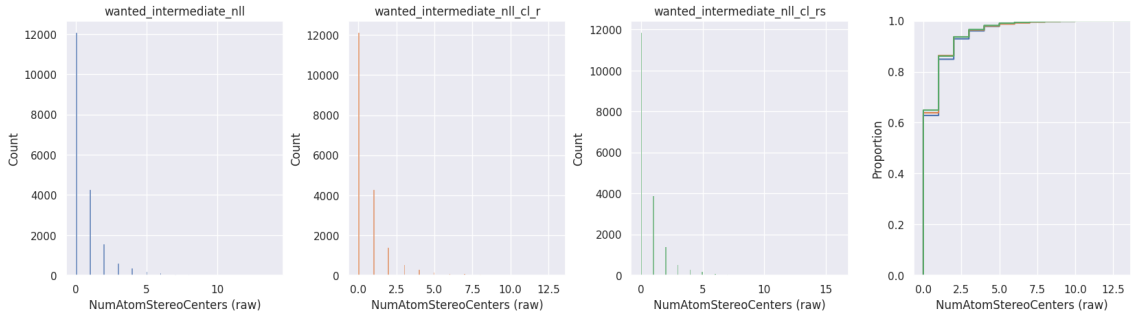


Figure A.57: Number of Atom Stereocenters

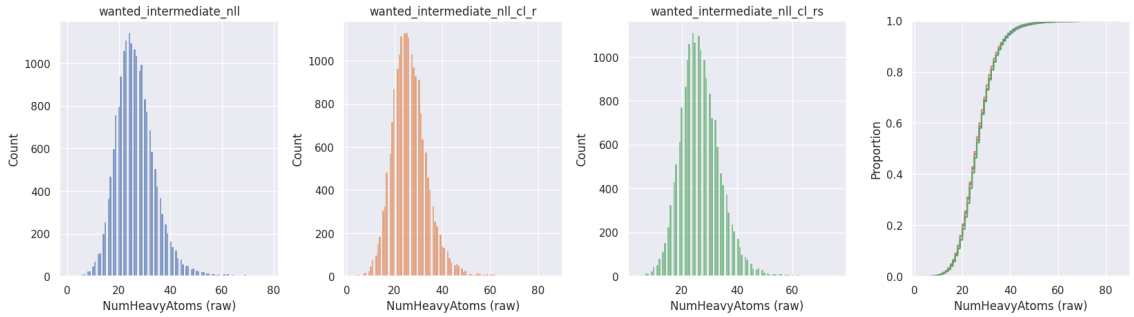


Figure A.58: Number of Heavy Atoms

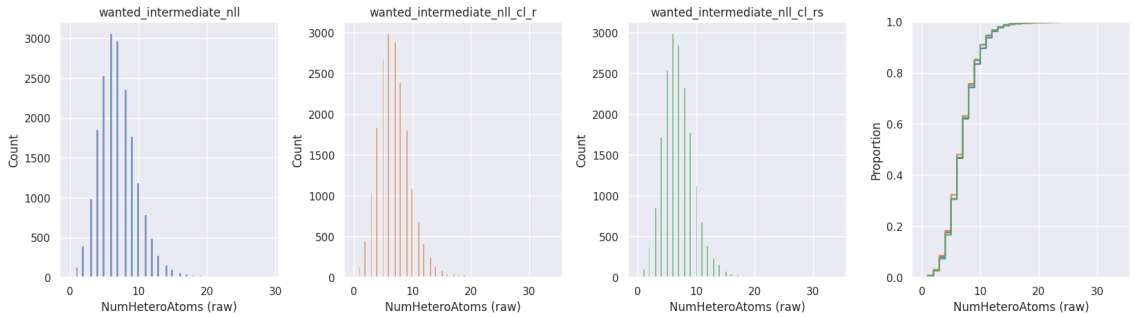


Figure A.59: Number of Hetero Atoms

Figure A.60: Number of Rings

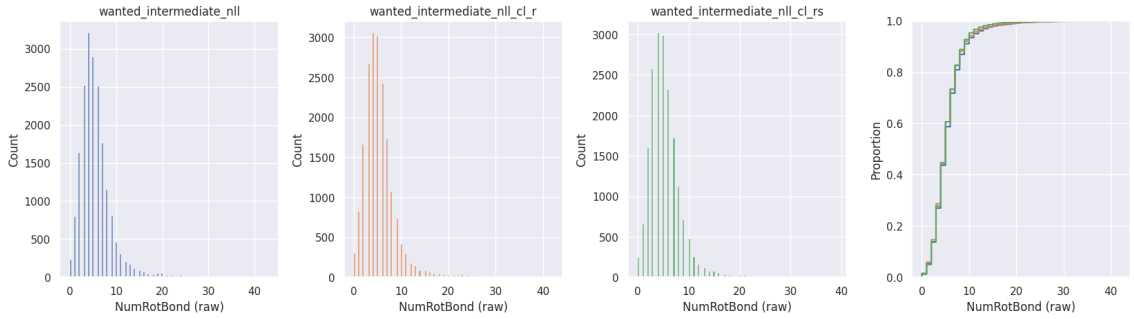


Figure A.61: Number of Rotatable Bonds

Figure A.62: sp Hybridization

Figure A.63: sp2 Hybridization

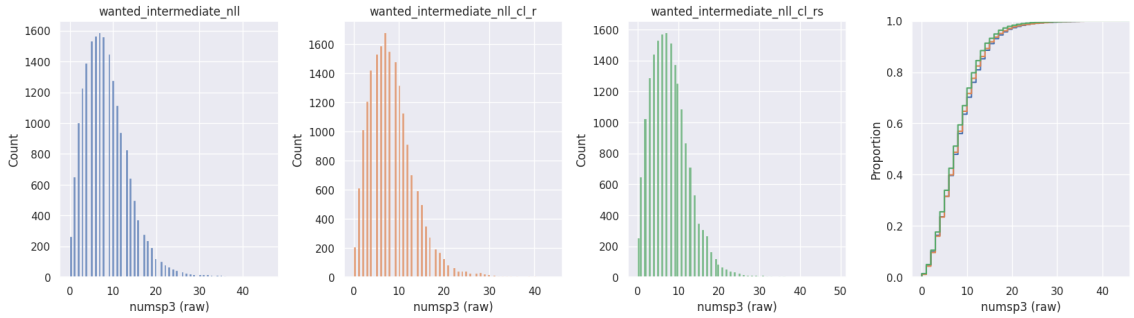


Figure A.64: sp3 Hybridization

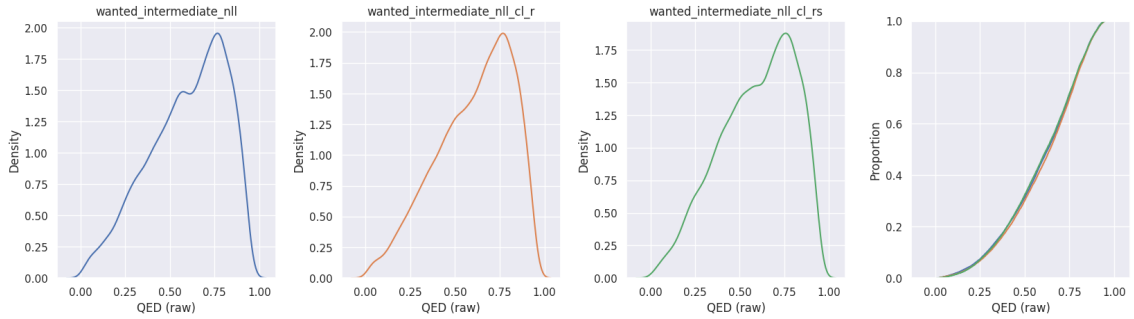


Figure A.65: Quantitative Estimate of Drug-likeness (QED)

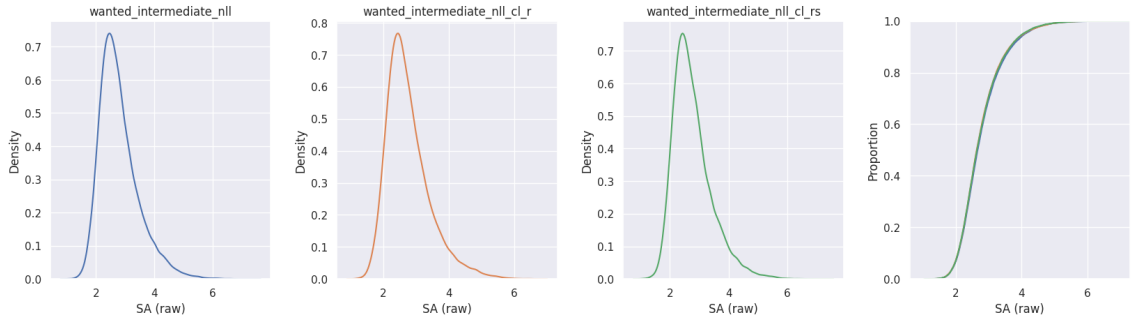


Figure A.66: Synthetic Accessibility

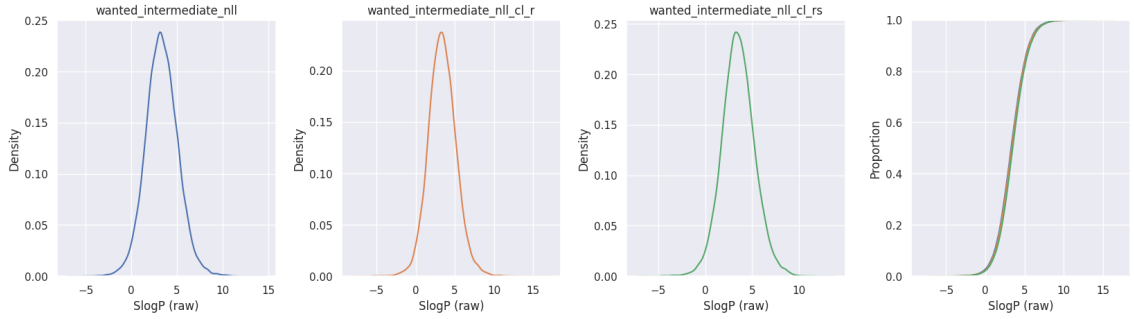


Figure A.67: SlogP (Octanol-Water Partition Coefficient)

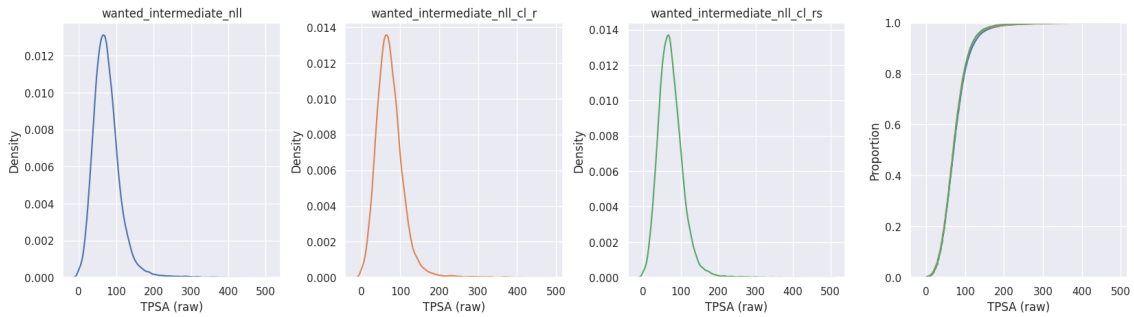


Figure A.68: Topological Polar Surface Area (TPSA)