Department of Computer Science
The University of Auckland
New Zealand

Machine Learning in the Generation of Novel Molecules

Charles David Douglas Tremlett

October 2019

Supervisors: *Dr Jörg Simon Wicker*
Dr Patricia Jean Riddle
Dr Jason Yan Cheuk Tam

A dissertation submitted in partial fulfillment of the requirements of
Master of Professional Studies in Data Science

Abstract

Techniques to sample from the vast space of possible molecular structures can aid in the design of novel chemicals with desired properties. Existing approaches rely on large databases ($n > 10^7$) and have high computational requirements. We demonstrate an efficient encoding of chemical space for a small data set of organic molecules known to undergo specific biotransformations ($n < 10^3$), leveraging techniques from NLP (natural language processing). We provide examples of the generation of semantically correct molecular representations.

Furthermore we show that the generated molecular representations can be assessed to determine whether they are likely to undergo specific biotransformations which exist in the initial data set.

Acknowledgements

I would like to acknowledge and thank:

- My wife Josephine and daughter Elsie, for their extensive patience and encouragement while I have undertaken this dissertation;
- My extended family for their consistent and ongoing support;
- My supervisors, for the excellent guidance they have provided.

Contents

1 Introduction

1.1 Motivation

The discovery of *de novo* chemical structures is an ongoing challenge in fields of drug discovery, pesticide and herbicide research, and others. In each of these fields novel molecules are sought which must have specific properties. For example, potential drugs are sought which have high affinity, selectivity, efficacy, metabolic stability, and oral bioavailability.

In drug discovery, more than 10,000 molecules may be synthesised and screened prior to one obtaining US FDA approval [19]. Estimates of the number of unique chemical structures in the restricted space of small organic molecules which are potentially pharmacologically active are in the region of 10^{63} [3]. This estimate applies limitations on the maximum molecular weight and only allows the use of five chemical elements (carbon, hydrogen, oxygen, nitrogen, and sulphur). Any improvement in the suitability of molecules in the candidate pool can contribute to an increased rate and decreased cost of drug discovery.

In pesticide and herbicide research, the prediction of microbial biotransformations of chemical structures remains a knowledge and time intensive task. Being able to predict microbial biotransformations is important to understand the likely breakdown and degradation of candidate chemicals in the environment. The traditional knowledge-based approach relies on expert knowledge of the basic sets of transformation rules, but can include rules which are too general, incomplete, or otherwise inconsistent.

Any chemical space with fewer restrictions (e.g. including the halogens) will be exponentially larger. Any systematic search or enumeration of such a space is computationally infeasible. Ruddigkeit et al. have created a database of all molecules of up to 17 atoms of C, N, O, S, and halogens with structures which are likely to exist [23]. This contains approximately 166.4 billion molecules. Applying a knowledge-based approach to even this limited data set would be incredibly time consuming, so any approach which may highlight candidate molecules likely to possess certain properties would benefit research.

1.2 Outline

We examine the University of Minnesota Biocatalysis/Biodegradation Database (UM-BBD) [5]. This dataset contains 712 unique molecules. Indications of 179 biotransformation rules ("btrules") are given for each molecule, whether the rule is triggered, is not triggered, or is not applicable.

We hypothesise that by finding an efficient encoding of molecules found in the UM-BBD, important features of the molecules triggering biotransformation rules will be preserved. From the encoded space, we will attempt to sample novel molecules which also trigger biotransformation rules.

Each molecule in the UM-BBD is given as a string using the simplified molecular-input line-entry system (SMILES) [28]. See section 2.1 *Working with SMILES* for more detail on this format.

There is naturally some scepticism as to whether SMILES strings can provide neural networks with a sufficiently accurate representation of a molecule for the network to learn to encode and reconstruct existing molecules, and then successfully generate new molecules. However, an equivalence can be drawn to natural language models, where text in language A can be encoded, and then decoded into a language B. A technique known as sequence-to-sequence (or "seq2seq") learning [26] can be used for tasks of this type, and we will exploit this to model SMILES strings.

1.3 Background

A microbial biotransformation is a modification made on a molecule by microorganisms or their products, such as bacteria, fungi, and enzymes. A comprehensive list of existing biotransformation rules can be found on the EnviPath website [30].

Extensive chemistry knowledge is not required to follow and understand the concepts of this paper. It is sufficient to understand that a if a molecule triggers a biotransformation, it can be altered by the addition or removal of atoms, or by the severing of bonds. However, we provide here a summary of common reactions, and will apply some understanding of chemistry in assessing novel molecules generated in section 6.2 *Samples of Molecules Triggering Biotransformation Rules.*

A sample of biotransformation rules assigned to reaction groups [6].

Reaction group	**btrules**
Alcohol oxidation	bt0001, bt0002
Aldehyde oxidation	bt0003
Aliphatic hydroxylation	bt0241, bt0242, bt0036, bt0332, bt0333, bt0334
Amide hydrolysis	bt0067, bt0318, bt0024, bt0027
Aromatic vic-diol ring cleavage	bt0008, bt0254, bt0041, bt0045, bt0069, bt0131, bt0165, bt0174, bt0184, bt0297
Aromatic ring dioxygenation	bt0005, bt0042, bt0055, bt0065, bt0072, bt0128, bt0196
Aromatic ring monooxygenation	bt0011, bt0012, bt0013
Phenolic ring monooxygenation	bt0014, bt0064
C=C bond reactions	bt0021, bt0049, bt0259, bt0291
CoA-thioester formation	bt0094, bt0315
Decarboxylation	bt0051, bt0082, bt0060, bt0072
Keto-ene hydrolysis	bt0040, bt0047
Keto-enol tautomerism	bt0231, bt0044
Oxidation of vic-di-H-di-OH to aromatic	bt0255, bt0056, bt0197

1.4 Related Work

There have been several interesting papers published which have applied machine learning algorithms to molecule discovery:

- Hybrid machine learning and knowledge based approaches have been shown to improve the accuracy when determining the applicability of biotransformation rules to chemical structures [29].

- Proof of concept work has shown that approaches using neural networks can screen chemical compounds for anti-cancer properties [9].

- A combined generative and predictive approach has been shown to successfully design chemical libraries satisfying a specific range of basic physical properties (e.g. melting point or hydrophobicity) [20].
- Recurrent neural networks have been used to generate molecules with biological activity against a specific target (e.g. malaria) [24].

2 Data Structure and Preparation

2.1 Working with SMILES

To understand what a SMILES represents, it is useful to consider the procedure for generating one from a molecular (or chemical) graph. A molecular graph represents the structure of a molecule through graph theory, where each of the vertices represents an atom, while each of the edges represents a chemical bond.

From the molecular graph, a SMILES is obtained by appending the symbols of nodes encountered in a depth-first tree traversal. The molecular graph must first be trimmed to remove hydrogen atoms and cycles should be broken to turn it into a spanning tree. Where cycles are broken, numeric labels are included as suffices to indicate which nodes are connected. Parentheses are used to denote branches on the tree.

Various choices made during the process can alter the resulting SMILES. Therefore an initial molecular graph can produce different, but equally valid SMILES. These choices include (a) which bond is chosen to break a cycle; (b) which atom is selected to start the traversal; and (c) the selection order of branches when they are encountered. We exploit this feature to augment our data set, the procedure for which is explained in section 3 *Data Augmentation* of this paper.

In the SMILES format, each non-hydrogen atom is represented by its chemical symbol (e.g. C, O, N, Cl). Simple bonds are denoted by - (single, although in practice usually omitted), = (double), # (triple), and $ (quadruple). Rings (cycles) which are broken to

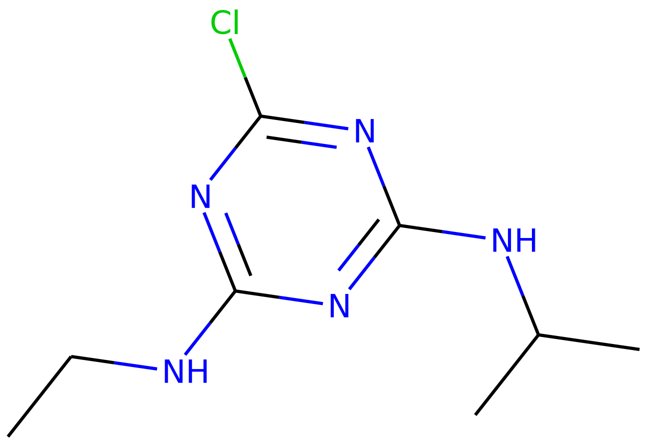


Figure 2.1: Two-dimensional representation of the herbicide Atrazine (6-chloro-N'-ethyl-N-propan-2-yl-1,3,5-triazine-2,4-diamine). This molecule has the SMILES `CCNc1nc(Cl)nc(NC(C)C)n1`.

produce the spanning tree are denoted by the suffix of a number to indicate the connection point (e.g. C1CCCCC1). Branches are described with parentheses (e.g. CCC(=O)O). Stereoisomers (which differ in the three-dimensional orientation of their atoms) can be specified by the use of \ or / to denote directional single bonds adjacent to a double bond (e.g. F/C=C/F and F/C=C\F are stereoisomers of 1,2-difluoroethylene).

Examining an example from the UM-BBD data set, the SMILES string CCNc1nc(Cl)-nc(NC(C)C)n1 represents Atrazine, a herbicide of the trizine class, which is widely used in the United States and Australia, but banned in the European Union for its persistence in groundwater. The IUPAC name for this molecule is 6-chloro-N'-ethyl-N-propan-2-yl-1,3,5-triazine-2,4-diamine.

In the UM-BBD, Atrazine is indicated as triggering the biotransformation rules bt0330 and bt0339, while it is not triggered by bt0029, bt0063, bt0333, and bt0334 (the remaining rules are not applicable). Each of these rules indicates a specific biodegradation pathway, for example the rule bt0039 represents an oxidative removal of a triazine amine side chain.

Using the open-source cheminformatics software package RDKit [14], we can draw and inspect the two-dimensional structure of this molecule. See figure 2.1 on page 6.

By examining the distribution of the lengths of SMILES in the UM-BBD data set, we can see that there is a bimodal distribution, see figure 2.2 on page 7. The length of

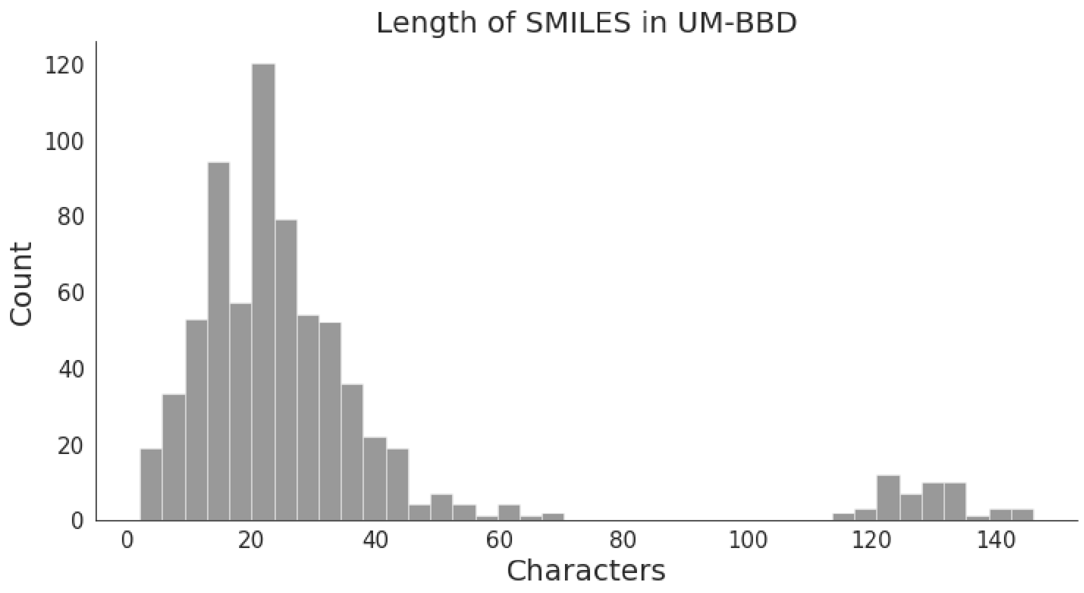


Figure 2.2: Distribution of SMILES length in the UM-BBD data set.

a SMILES does not correspond exactly to the size of the molecule (number of atoms), however there is a strong relationship, see figure 2.3 on page 8. This discrepancy exists because different structures are represented in a SMILES using a different number of characters, even though they might contain the same number of atoms.

Rather than attempting to design and train a model which would map molecules from both the identified distributions to a hidden space, we have elected to remove the molecules with a SMILES length greater than 50. We have filtered the data set to retain only those molecules with a SMILES length of 50 or below. This leaves us with 646 molecules from the original UM-BBD data set (90.7%).

2.2 Preparation for Autoencoder

Working with sequences of data requires a recurrent neural network. As such we need to prepare molecules in the UM-BBD data set for the first layer of the autoencoder, a LSTM (Long Short-Term Memory) network layer.

The dimensions of the input data are (n, l, c), where n is the number of samples, l is the maximum length of the sequences in the data set, and c is the number of unique characters in the data set. One SMILES (a sequence of characters) is one sample, one time step is one point of observation in the sample, and one feature is one observation at a time step.

In the SMILES representing the 646 molecules in the UM-BBD data set, there are 31 unique characters, or symbols.

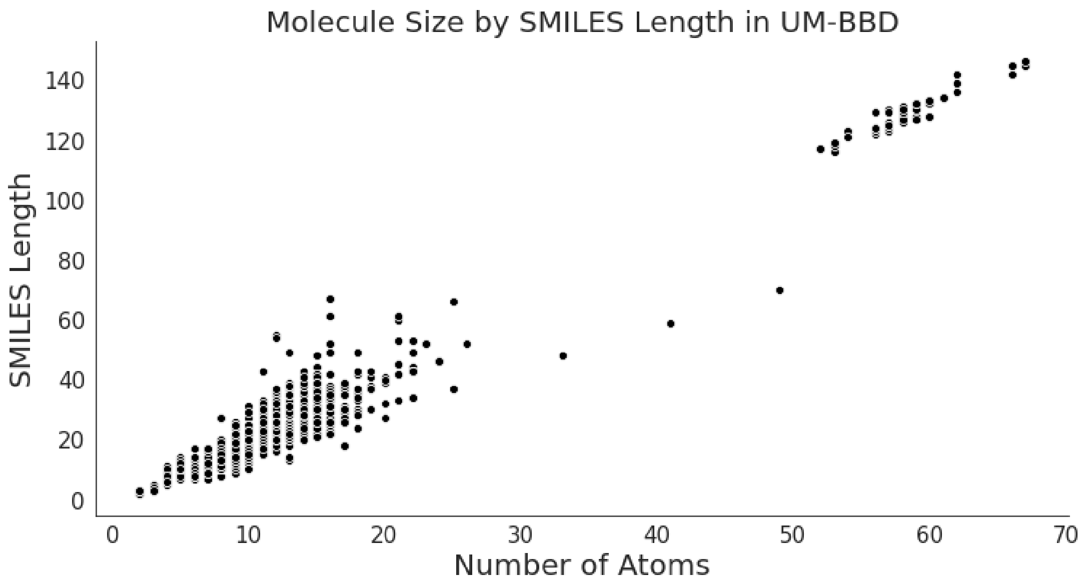


Figure 2.3: Relationship between the number of heavy atoms in a molecule, and the length of the associated canonical SMILES in the UM-BBD data set.

Frequency of characters in SMILES in the UM-BBD

Character	**Frequency**	**Character**	**Frequency**
c	2859	3	117
C	2621	n	80
O	1507	/	59
(	1259	4	48
)	1259	$+$	48
1	936	S	45
$=$	866	5	28
]	523	B	28
[	523	r	28
$-$	360	F	15
2	336	s	13
l	229	P	9
N	204	#	8
@	173	6	2
\	171	o	1
H	120		

We have used the character E to pad all SMILES shorter than 50 characters. We require that each sequence ends with an E, denoting the end of a SMILES sequence. We also use a ! in the first position to create an offset for teacher forcing, see section 4.4 *Teacher Forcing*. This brings the total number of unique characters to 33, and the maximum length is 50.

Note that some atoms are represented by two characters (e.g. Cl for chlorine), and here we have not tokenised these as its own token. By doing this we, in essence, expect the model to learn which combinations of characters can go together (e.g. that an l can follow a C). An improvement might be to tokenise these specific atoms as separate tokens (in our data set this is only Cl and Br), however we do not expect this to lead to a significant improvement in the model.

Some atoms can be represented by either a lower case or upper case character. Lower case characters are used in the case of aromatic structures, which contain a ring of resonance bonds that produce highly stable molecules.

For each character in a SMILES, we apply one-hot encoding to produce a vector of length $c = 33$ (the number of unique characters in our data set) such that the position where 1 appears identifies the character. For example, the following vector may identify a C:

$$\begin{pmatrix} 0 & 1 & 0 & \cdots & 0 \end{pmatrix} \tag{2.1}$$

This procedure is repeated for each character in the SMILES to produce a sparse matrix of dimensions $l \times c = 50 \times 33$. This represents a single complete SMILES.

Summing up the above, the dimensions of our input data are given by $(n, l, c) = (646, 50, 33)$.

After data augmentation (see section 3 *Data Augmentation*), we have 91,299 samples. We separate these into training and testing data sets using random shuffling. Given the number of samples we have, we have elected to use 80% of the data as training (73,039 samples), 16% as validation or testing (14,608 samples), and 4% as a hold out set (3,652 samples).

2.3 Preparation for Discriminator

For the discriminator, we relied on an embedding known as mol2vec [8]. An embedding is a mapping of categorical data to a continuous vector representation in a continuous space of n dimensions (i.e. $\mathbb{R}^n$). In the case of mol2vec, the categorical information to be mapped consists of identifiers of substructures of molecules.

The mol2vec embedding is 300-dimensional, and has been trained on a set of 20 million molecules from the ZINC database of commercially available molecules used for virtual screening [25].

There are several different schemes for generating unique identifiers for substructures in molecules. Here mol2vec uses Extended Connectivity Fingerprints (ECFPs) [22]. The benefit of using ECFPs is that they have been developed for activity modelling, rather than similarity searching as previous iterations of topological fingerprints have.

We began by converting the 646 SMILES in the UM-BBD to an object of the `Mol` class using RDKit. These Mol objects contain an internal representation of the expected atoms and bonds, and the class belongs to the RDKit project.

We then convert the `Mol` objects to ECFP fingerprint sequences using the Morgan algorithm [17]. This algorithm allows us to specify the number of steps to adjacent atoms to consider when generating the fingerprint, and this is known as the radius. We used a radius of four. Each of the fingerprint sequences for each molecule were then converted to a 300-dimensional vector using the mol2vec embedding.

We considered the use of the mol2vec embedding in training the autoencoder, however there is no straightforward method of reversing the generating of the ECFP fingerprint sequences, due to the use of a hash function in their calculation. In any case, it strengthens our results to have used completely independent methods of constructing the generator and the discriminators.

Note that we cannot use augmented data to train the discriminator (see section 3 *Data Augmentation*). This is because the various non-canonical SMILES are represented identically to their canonical SMILES in the mol2vec embedding (i.e. they are represented by identical vectors).

The 646 SMILES in the training set were separated into training (80%, 517 samples) and test sets (20%, 129 samples). Given the low number of positive samples for each biotransformation rule, it was essential to apply stratification of the classes when splitting the data into training and testing sets.

3
Data Augmentation

Data augmentation refers to the technique of extending an existing data set by altering objects in that data set in some way. The most common use of data augmentation in machine learning is in problems related to image processing, where transformations are applied to images in order to improve the generalisation of the model. Examples of transformations of images include rotation, mirroring, cropping, and alterations to the colour. It has been shown to be highly successful in machine learning problems which require a model to identify objects in images.

Given the limited data set in the UM-BBD (712 molecules, of which we are working with 646), we explored whether a similar technique could be used to augment this data set. As short text strings, SMILES cannot be altered arbitrarily without changing the structure of the molecule. For example, adding an extra atom of any type changes the structure of the molecule, and we cannot know whether it has the same properties as the original molecule (i.e. whether it triggers the same biotransformation rule).

However, the same molecule can be represented by multiple equally valid SMILES strings. One is chosen algorithmically from an internal representation of the molecular structure as the canonical SMILES.

For example, the substance chloroacetaldehyde is represented canonically by the SMILES string ClCC=O, but can also be represented by the equally valid string C(Cl)C=O, or any one of several other SMILES strings. See figure 3.1 on page 12.

In order to identify valid SMILES strings we leveraged a feature in RDKit which allows

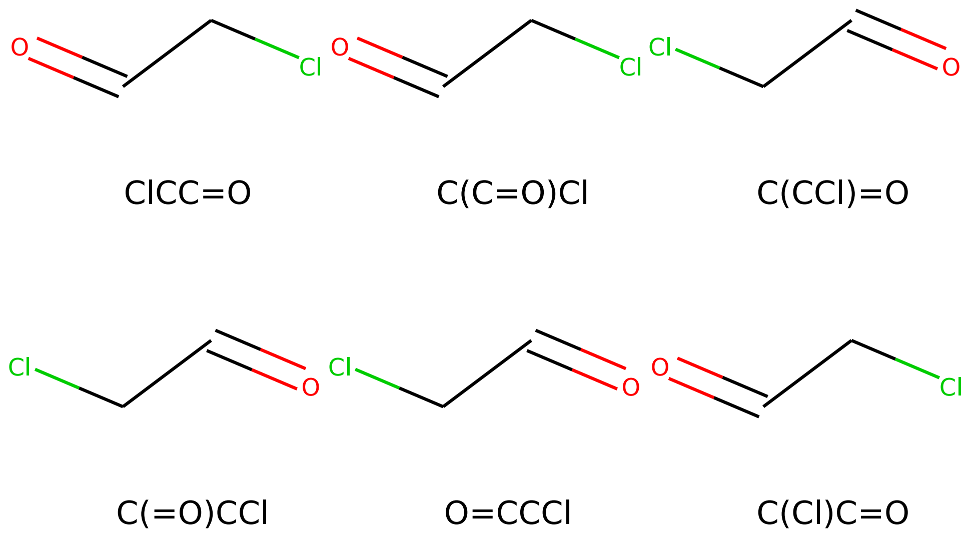


Figure 3.1: Various possible SMILES representing chloroacetaldehyde.

atoms in a `Mol` object to be reordered, and then converted to a SMILES without forcing its canonicalisation. The pseudo-code in algorithm 1 illustrates the procedure we followed to generate 90,881 additional SMILES which represent the same molecules as the 646 in the UM-BBD data set.

Algorithm 1 Pseudo-code for SMILES augmentation

```
$n$ = number of times to shuffle the atoms
**for** each SMILES in data set **do**
    $i = 0$
    Convert SMILES string to RDKit Mol format
    Get number of atoms in the Mol object
    Create an array with a range from zero to the number of atoms
    **while** $i < n$ **do**
        Randomly shuffle the array
        Renumber atoms in the Mol object
        Convert Mol object to SMILES without canonicalisation
        $i = i + 1$
    **end while**
**end for**
```

Considering the example of chloroacetaldehyde, there are four atoms, so there are 4!, or 24 permutations. However, since two of these atoms are carbon (C), and can be considered identical, the actual number of unique permutations is reduced to 12. Then, due to the algorithm which generates SMILES from the `Mol` object, there are finally six unique SMILES which represent the same molecule.

Given the step where the atom numbering is randomly shuffled, this does not guarantee the same generated SMILES strings every time. Because we may shuffle atoms into the same order as a previous iteration, this is not an optimised solution. It may be better to calculate all permutations of the number of atoms. However this would be computationally expensive, indeed impossible for all but the smallest of n. The longest SMILES in our sample (50 characters), would have $50! \approx 3 \times 10^{64}$ possible permutations.

Given the embarrassingly parallel nature of the shuffling of the atoms for SMILES augmentation and the multi-core capability of modern CPUs, we parallelised the algorithm above in order to increase the number of sampled molecules in a given time.

4 Autoencoder Design

4.1 Autoencoder Outline

An autoencoder is an unsupervised artificial neural network trained to find an efficient encoding for a given data set. Initially used in dimensionality reduction and feature learning, they have recently been applied in generative models [12].

While it is trained as a single network, an autoencoder is best thought of as two separate components, an encoder network, and a decoder network.

The trained encoder network takes a data point $x \in X$ as input, and maps it to a vector $z \in Z$. It can be defined by the function ϕ:

$$\phi : X \to Z \tag{4.1}$$

The latent space Z (also known as the hidden space) is a n-dimensional space in which each of the data points is encoded as a vector of length n. The dimension of the space, n, is a parameter which is given prior to the training of the autoencoder.

The decoder network takes an encoded vector $z \in Z$, and maps it to a data point $x' \in X$. It can be defined by the function ψ:

$$\psi : Z \to X \tag{4.2}$$

The objective of the autoencoder is to minimise the difference between the original x

and the reconstructed x'. That is, the reconstructed x' should be as close as possible to the original x. A simple squared error loss function $\mathcal{L}$ can be defined:

$$\mathcal{L}(x, x') = \|x - x'\|^2 \tag{4.3}$$

Or defined solely in terms of x:

$$\mathcal{L}(x) = \|x - \psi(\phi(x))\|^2 \tag{4.4}$$

A squared error loss function is commonly used in statistical models, often without consideration of the actual loss in the application. The use of the squared error as a loss function was first proposed by Gauss in 1821, but he acknowledged it as arbitrary. [15] Its use as a default loss has been criticised more recently [2], however the discussion on the appropriateness of the squared error loss function is outside the scope of this paper.

In any case, our data set is not suited for a squared error loss function, because we are not modelling a continuous function. We are modelling the output from a finite set of categorical variables. Each step in the sequence which makes up each sample belongs to one and only one of c classes. That is, the target vector is a one-hot vector with a single positive class and $c - 1$ negative classes.

As we are working with a multi-class classification problem, we have applied a cross-entropy (log) loss. Cross-entropy loss penalises incorrect classifications made with high confidence more strongly than those with low confidence. For a single sample in our data set, this can be defined mathematically as

$$\mathcal{L}(x) = -\sum_{i}^{L}\sum_{j}^{C} x_{i,j}\log(p_{i,j}) \tag{4.5}$$

where C is the number of classes, and L the number of observations. $x_{i,j}$ gives the correct label (either 0 or 1) for the j-th class in the i-th position in the sequence. $p_{i,j} \in (0,1) : \sum_j p_{i,j} = 1 \forall i, j$ gives the probability predicted by the model for the j-th class in the i-th position in the sequence.

4.2 Encoder

To construct the autoencoder, we will leverage work which has been done on sequence-to-sequence learning in the field of language translation [26].

All models described in this paper were implemented using the Keras framework [4] with a Tensorflow backend [1].

The encoder begins with a single layer of LSTM cells to read the input. Long Short-Term Memory cells are designed to keep an internal state over multiple steps in a recurrent neural network. They replace standard recurrent units and improve the performance of the network in recognising interactions separated by multiple time steps.

There is no hard and fast rule, or even a rule of thumb, for setting the number of units (neurons) in intermediate layers in a neural network. The basic principle which should be respected, is that there should be sufficient units for the network to learn complex relationships, but not so many that it can begin to memorise specific instances from the training data. These scenarios are analogous to the concepts of under-fitting and over-fitting. We must find an optimum value for the number of units which provides an accurate output, but also generalises well to the validation and holdout data sets.

We therefore experimented with the number of LSTM cells in this layer: 2, 4, 8, 16, 32, 64, 128, and 256. An increased number of cells can increase model accuracy as the network learns more complex relationships, but can increase the risk of overfitting, and significantly increases training time.

We configured the LSTM cells to return the internal h and c states. These states are then fed to a single fully connected (dense) layer with the number of units equivalent to the number of dimensions we desire in the latent space.

4.3 Decoder

Subsequently, the output from the latent space is then fed through two fully connected (dense) layers to decode the internal states to be set on a further LSTM layer. The number of units in these dense layers is equal to the number of units in the subsequent LSTM layer. This LSTM layer also receives the original input again, so it can be tasked with predicting the next character in the sequence. (This is a technique known as teacher forcing, see section 4.4 *Techer Forcing*.) A final fully connected (dense) layer with $c = 33$ units and softmax activation predicts the character.

4.4 Teacher Forcing

Rather than use the output from the previous recurrent step ($X(t-1)$) as input for the model to predict the current step ($X(t)$), we use the previous character from the original input vector ($y(t-1)$). This forces the decoder to continue to learn relevant information, even if it makes a mistake near the beginning of the sequence. This technique is known as teacher forcing [31].

A complete network diagram, including the encoder, decoder, and teacher forcing components can be found in figure 4.1 on page 18

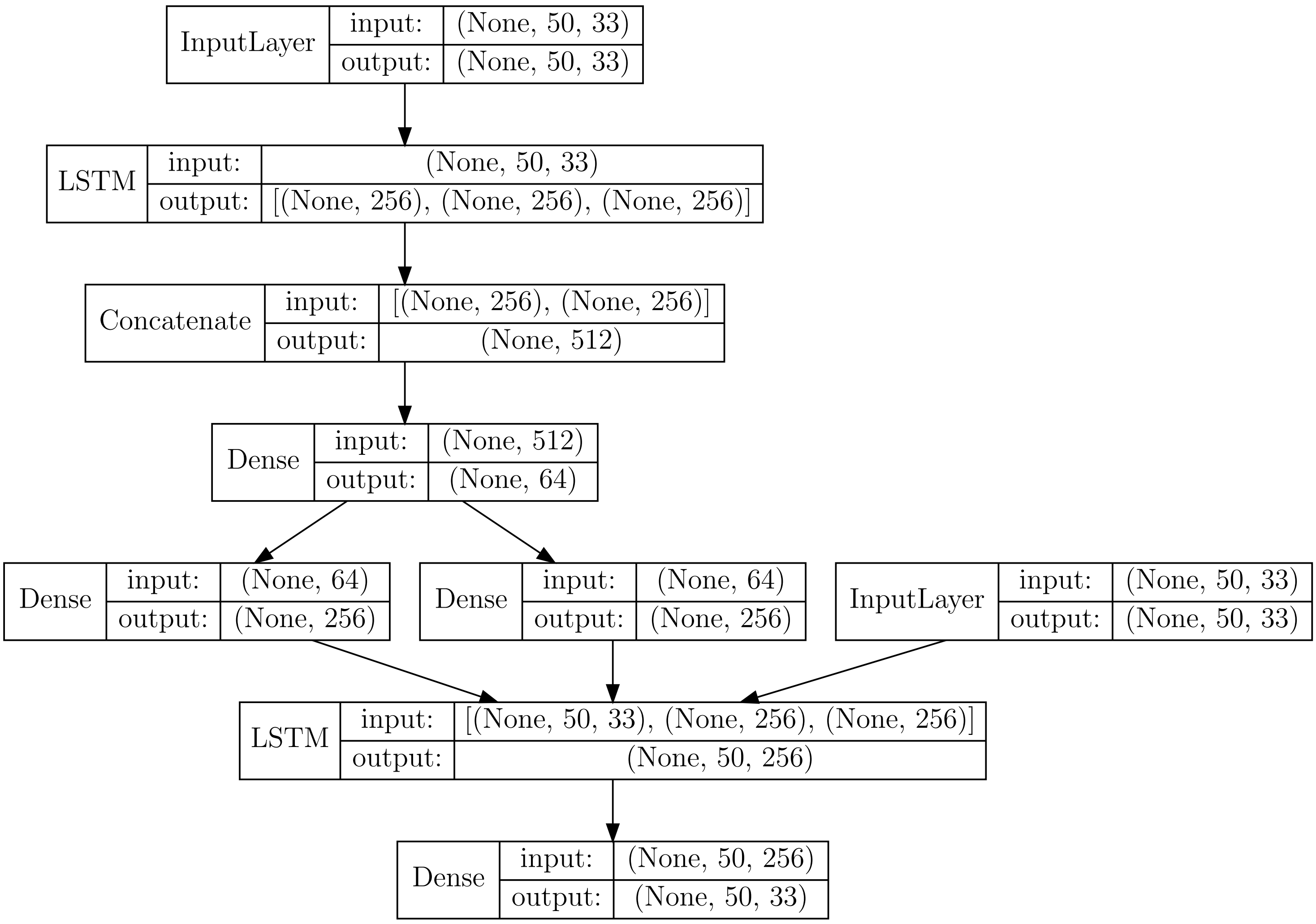


Figure 4.1: Illustration of the autoencoder network structure.

4.5 Model Training

A lower number of dimensions in the latent space reduces the risk of overfitting. We experimented with dimensions including 4, 8, 16, 32, and 64, and identified that a 64-dimensional hidden space allowed an optimal reconstruction of hold-out molecules.

The activation SELU (Scaled Exponential Linear Units) [13] was used for all fully connected layers, except for the final layer with softmax activation. Experiments showed that using SELUs as the activation functions produced better results than either ReLU, LeakyReLU or tanh.

Our model was trained over 100 epochs, using an Adam optimiser [11] with an initial learning rate of 0.005. The learning rate was halved if the loss of the validation data set did not decrease over five consecutive epochs. Early stopping was applied if the loss of the validation data set did not decrease over ten consecutive epochs. The batch size applied was 32 for the original data set, and 512 for the augmented data. In training a neural network, there is a trade off between batch size and number of epochs required.

The loss used was categorical cross-entropy.

In the following table we present a subset of our experimental results to illustrate our best and near-best findings. Complete tables of the results of our experiments can be found in the appendices.

Latent		Original Data			
Dimensions	LSTM Units	32	64	128	256
4	Training loss [a]	0.274	0.155	0.170	0.140
	Validation loss [b]	0.325	**0.294**	0.334	0.333
	SMILES reconstruction [c]	0.0%	3.8%	0.0%	0.0%
	Character reconstruction [d]	89.9%	91.0%	88.7%	90.5%
32	Training loss [a]	0.242	0.184	0.190	0.180
	Validation loss [b]	0.307	0.304	0.338	0.335
	SMILES reconstruction [c]	**11.5%**	0.0%	0.0%	0.0%
	Character reconstruction [d]	**91.8%**	90.1%	88.9%	88.5%

Latent		Augmented Data			
Dimensions	LSTM Units	32	64	128	256
16	Training loss [a]	0.284	0.105	0.018	0.005
	Validation loss [b]	0.290	0.113	0.035	0.022
	SMILES reconstruction [c]	1.2%	13.5%	61.9%	76.5%
	Character reconstruction [d]	89.4%	95.8%	98.9%	99.4%
64	Training loss [a]	0.253	0.118	0.004	**0.000**
	Validation loss [b]	0.258	0.125	0.015	**0.006**
	SMILES reconstruction [c]	1.3%	11.2%	84.9%	**94.9%**
	Character reconstruction [d]	90.6%	95.3%	99.6%	**99.9%**

[a] The loss evaluated against the training data set after 100 epochs.
[b] The loss evaluated against the validation data set after 100 epochs.
[c] The percentage of SMILES in the holdout data set which were completely reconstructed.
[d] The percentage of characters in SMILES in the holdout data set which were successfully reconstructed.

Figure 4.2 on page 20 illustrates the training and validation losses for the best model (with 64 dimensions in the latent space, and 256 units in the LSTM layers).

Note that during model training, an increasing divergence between the training loss and the validation loss indicates that the model is overfitting. This was observed in all experiments with the original (non-augmented) data set, suggesting that either we had insufficient data, or were using too many LSTM units.

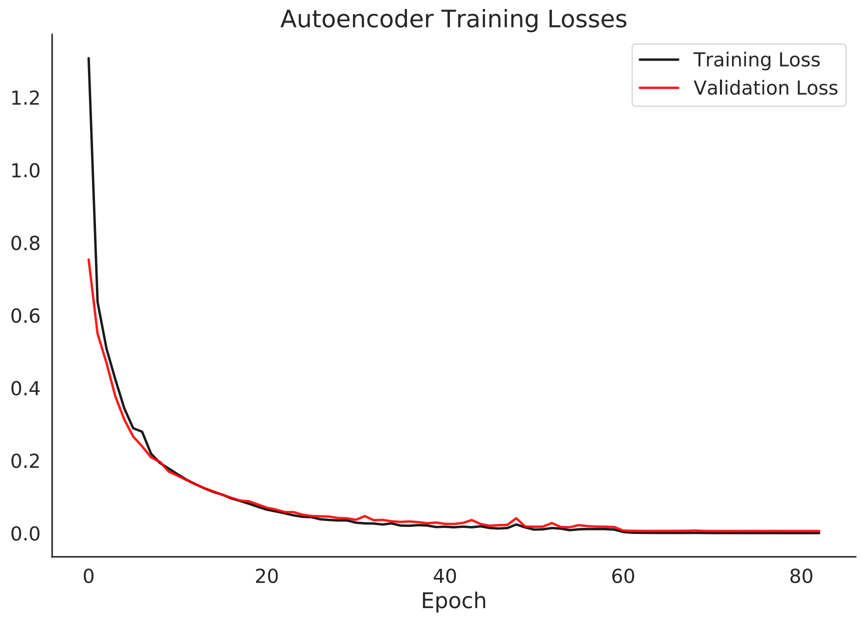


Figure 4.2: Training and validation loss for the autoencoder.

4.6 Exploring the Latent (Hidden) Space

We can visualise the latent space by performing further dimensionality reduction. Remembering that the latent space of our best model is our best effort in representing the molecules in a 64-dimensional space, we can further reduce this to two dimensions. Common techniques for performing dimensionality reduction for visualisation include PCA [7], T-SNE [27], and UMAP [16].

4.6.1 Principal Component Analysis (PCA)

Principal Component Analysis takes a set of observations of possibly correlated variables and transforms them into a set of values of linearly uncorrelated variables. As the first principal component accounts for the largest amount of variability in the data as possible, and the second the second most, the first two principal components can be plotted to give a view the relationship between molecules triggering different biotransformation rules in the data set.

In figure 4.3 on page 21 we have plotted the first two principal components for the four most common biotransformation rules. Note the areas which are dominated by one rule. This suggests that our autoencoder has been able to learn some distinctive characteristics of the molecules which trigger each rule. The first two principal components here capture 60.4% of the variance in this the subset of molecules triggering the four given rules.

For reference, the first five and ten principal components capture 80.4% and 92.5% of

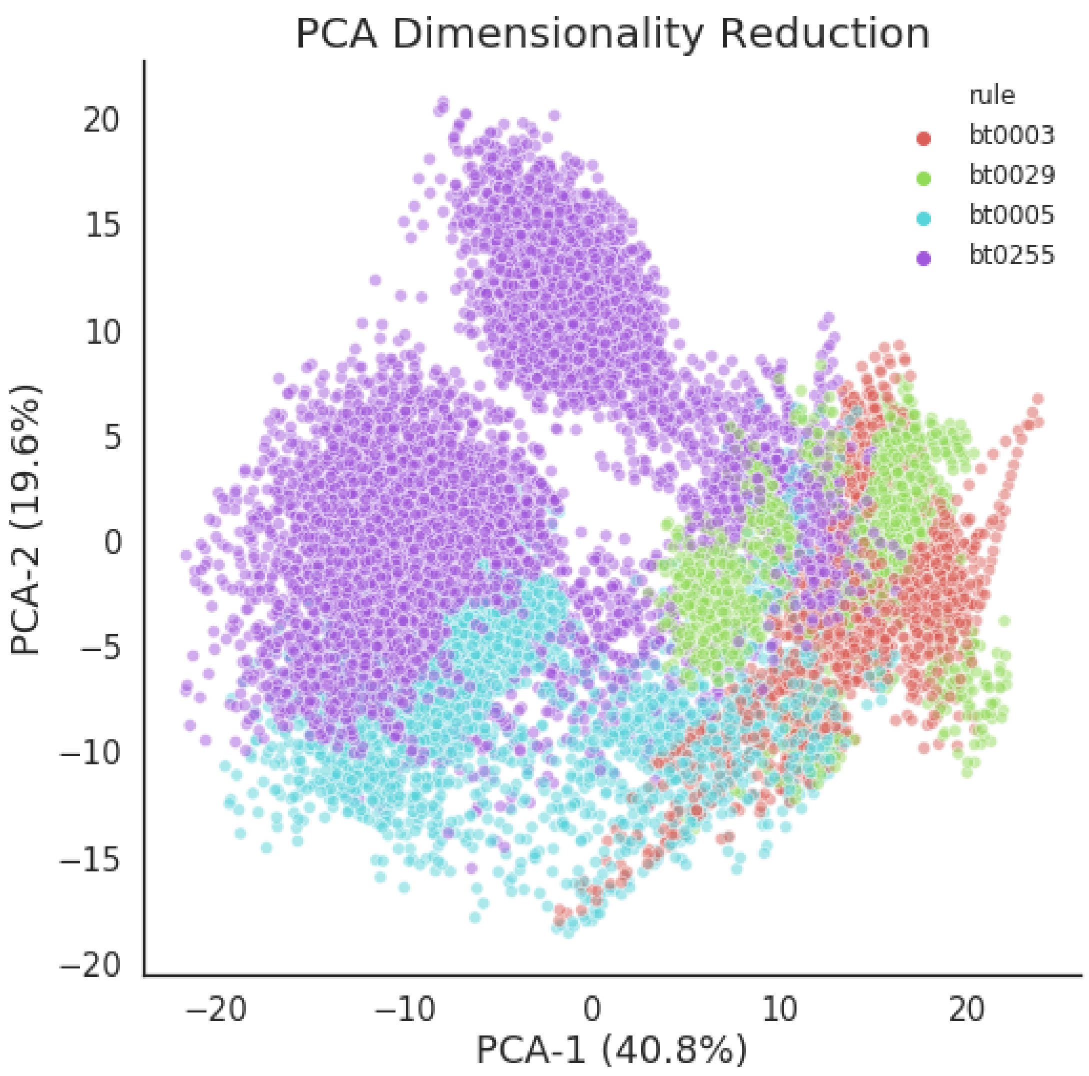


Figure 4.3: Principal Component Analysis dimensionality reduction for SMILES representing molecules which are triggered by four biotransformation rules. Regions are clearly separated in many areas of the plot. Note the coverage percentage for each of the first two principal components, 40.8% and 19.6% respectively. The plot captures 60.4% of the variance in the latent space between the set of molecules triggered by these four biotransformation rules.

the variance, respectively.

PCA is a venerable method of dimensionality reduction, being discovered by Karl Pearson in 1901 [7]. As such, it has limitations, and it has been superseded by nonlinear manifold learning methods such as t-SNE and UMAP.

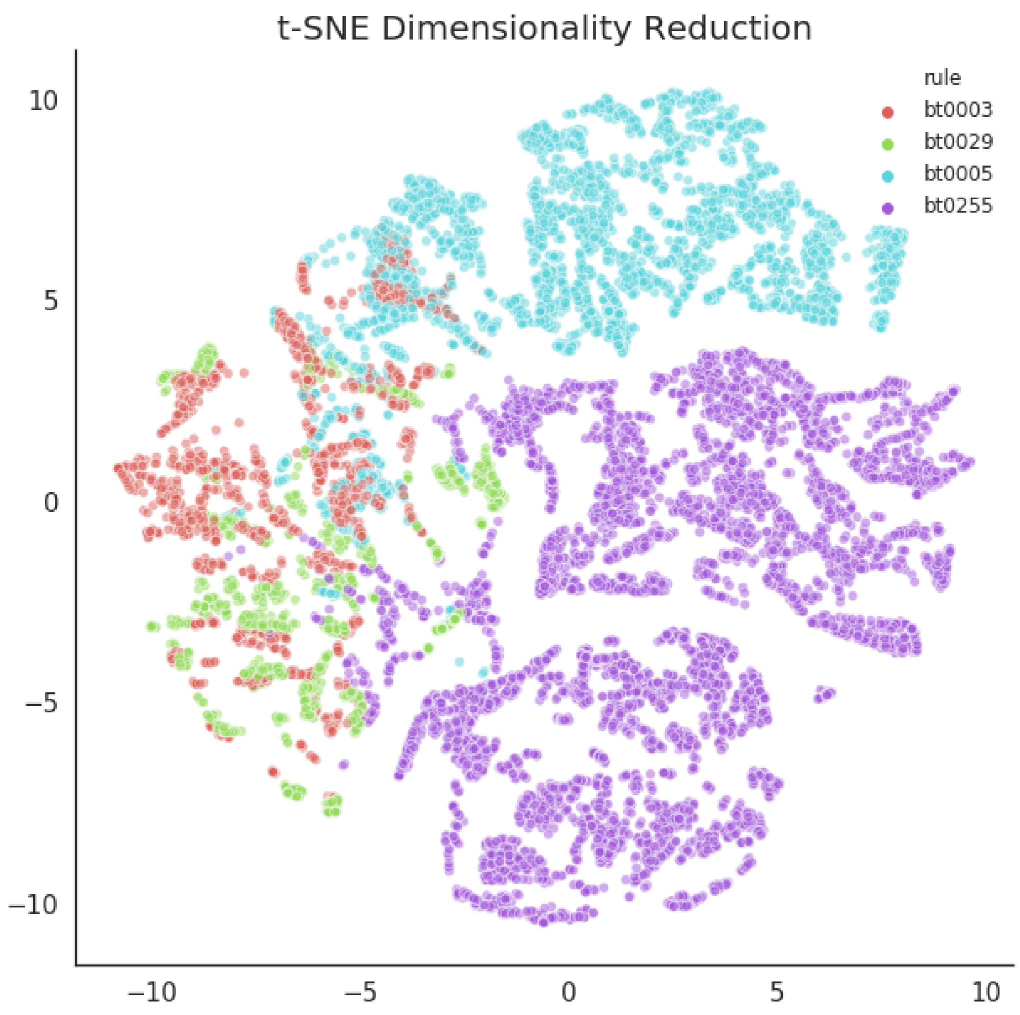


Figure 4.4: t-distributed Stochastic Neighbor Embedding dimensionality reduction for SMILES representing molecules which are triggered by four biotransformation rules.

4.6.2 t-distributed Stochastic Neighbor Embedding (t-SNE)

t-SNE is a nonlinear manifold learning method. It attempts to map high-dimension data to a low-dimension manifold, creating an embedding which should maintain local structures within the data as much as possible.

In figure 4.4 on page 22 we have plotted the T-SNE embedding for the four most common biotransformation rules. The delineation between the molecules triggering bt0005 and bt0255 is extremely clear, while there is more overlap between those triggering bt0029 and bt0003.

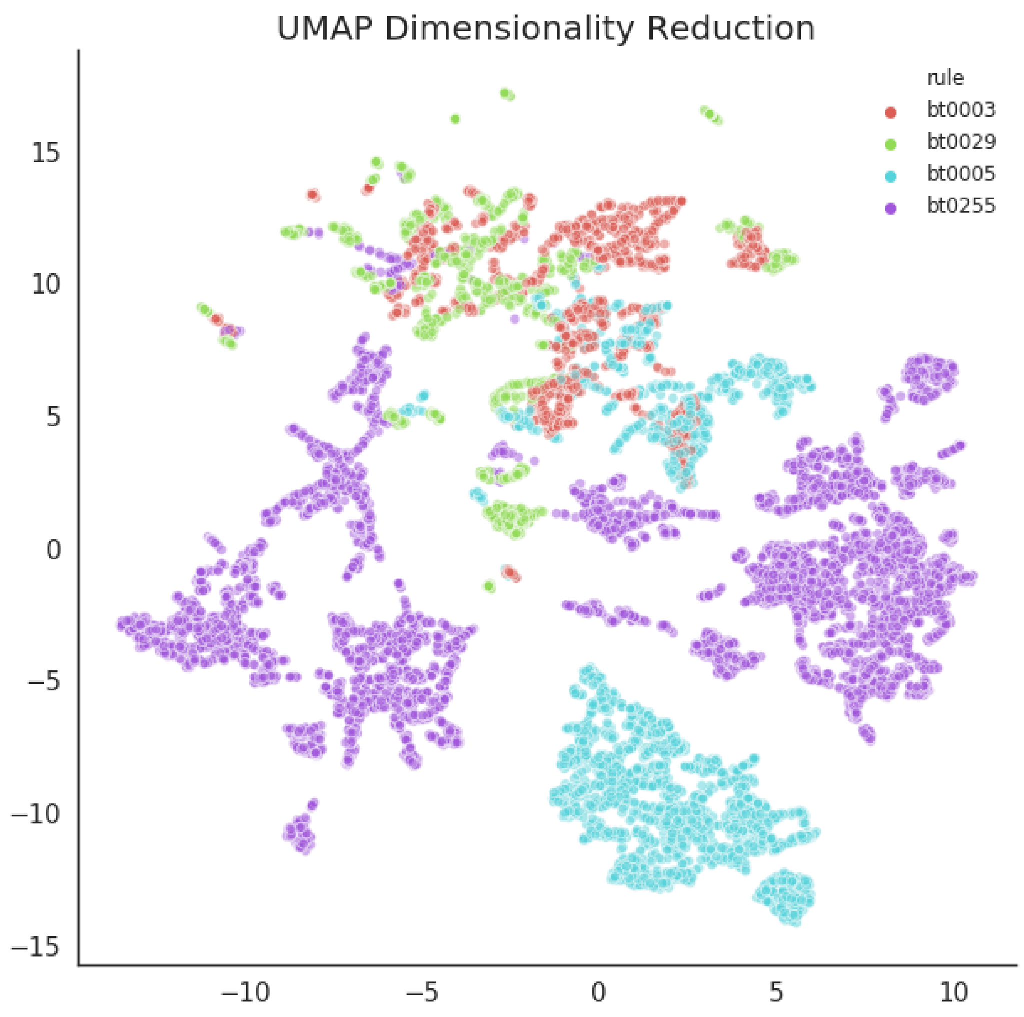


Figure 4.5: Uniform Manifold Approximation and Projection dimensionality reduction for SMILES representing molecules which are triggered by four biotransformation rules.

4.6.3 Uniform Manifold Approximation and Projection (UMAP)

UMAP is another nonlinear manifold learning method. It is similar to t-SNE, except it assumes that the data is uniformly distributed on a locally connected Riemannian manifold. UMAP is a very new method of dimensionality reduction, but provides an interesting alternative to t-SNE for visualisation.

In figure 4.5 on page 23 we have plotted the UMAP embedding for the four most common biotransformation rules.

5 DiscriminatorDesign

Our discriminator model seeks to identify molecules which trigger a given biotransformation rule. The approach we have taken uses a neural network with both convolutional and fully connected layers. Furthermore, we use the pre-trained embedding layer "mol2vec" to represent molecules as dense vectors.

Given the limited data set we are working with, we apply a strong regularisation layer with a dropout factor of 0.5.

An illustration of the discriminator network structure can be found in figure 5.1 on page 26.

5.1 Discriminator Training

The small number of positive samples for each biotransformation rule means that the discriminator model can be heavily influenced by the structure of the molecules in the training data set. In section 7.2.3 *Augment Data Set for Training Discriminator* we propose a method of augmenting the data set to increase the ability of the model to generalise.

For the twelve biotransformation rules with the most positive samples, the training and validation losses are plotted in figure 5.2 on page 27.

The loss calculated after each epoch was binary crossentropy. The batch size for training was set at eight samples, and each model was trained for five epochs.

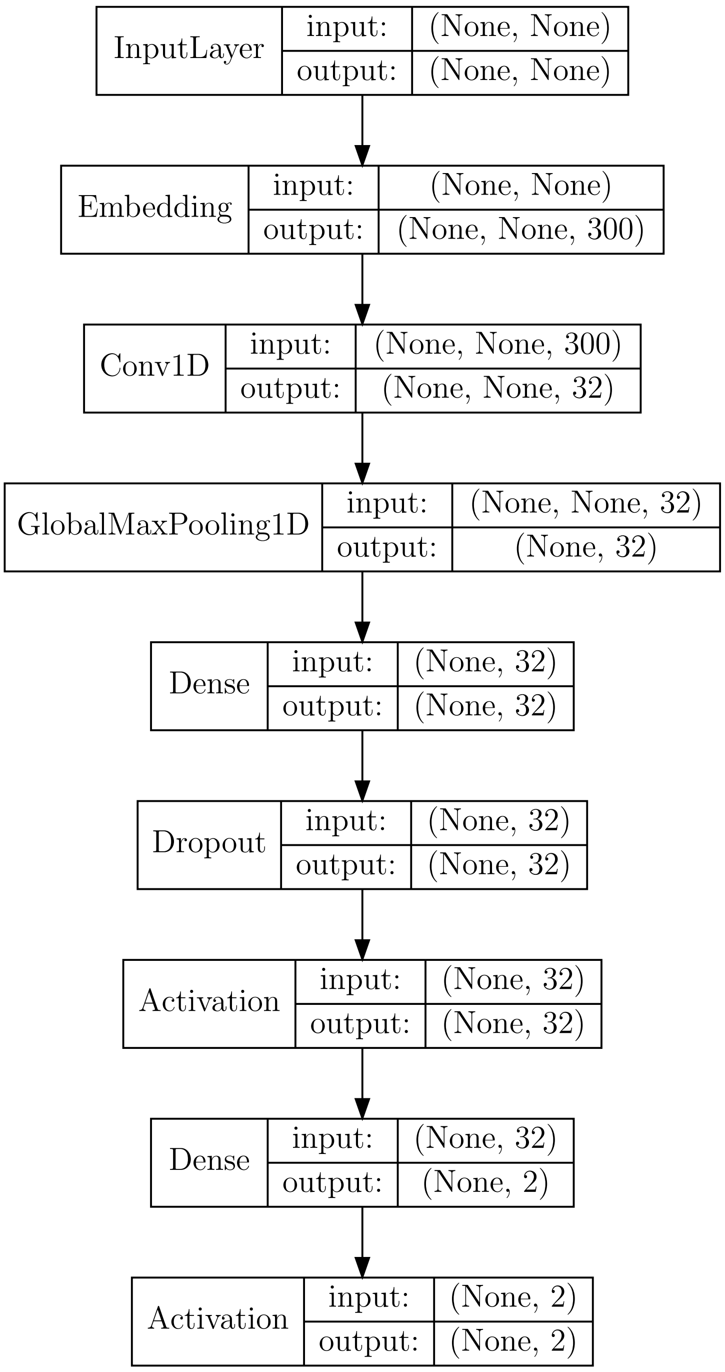


Figure 5.1: Illustration of the discriminator network structure. Note that the same structure was used to build a separate model for each of the biotransformation rules.

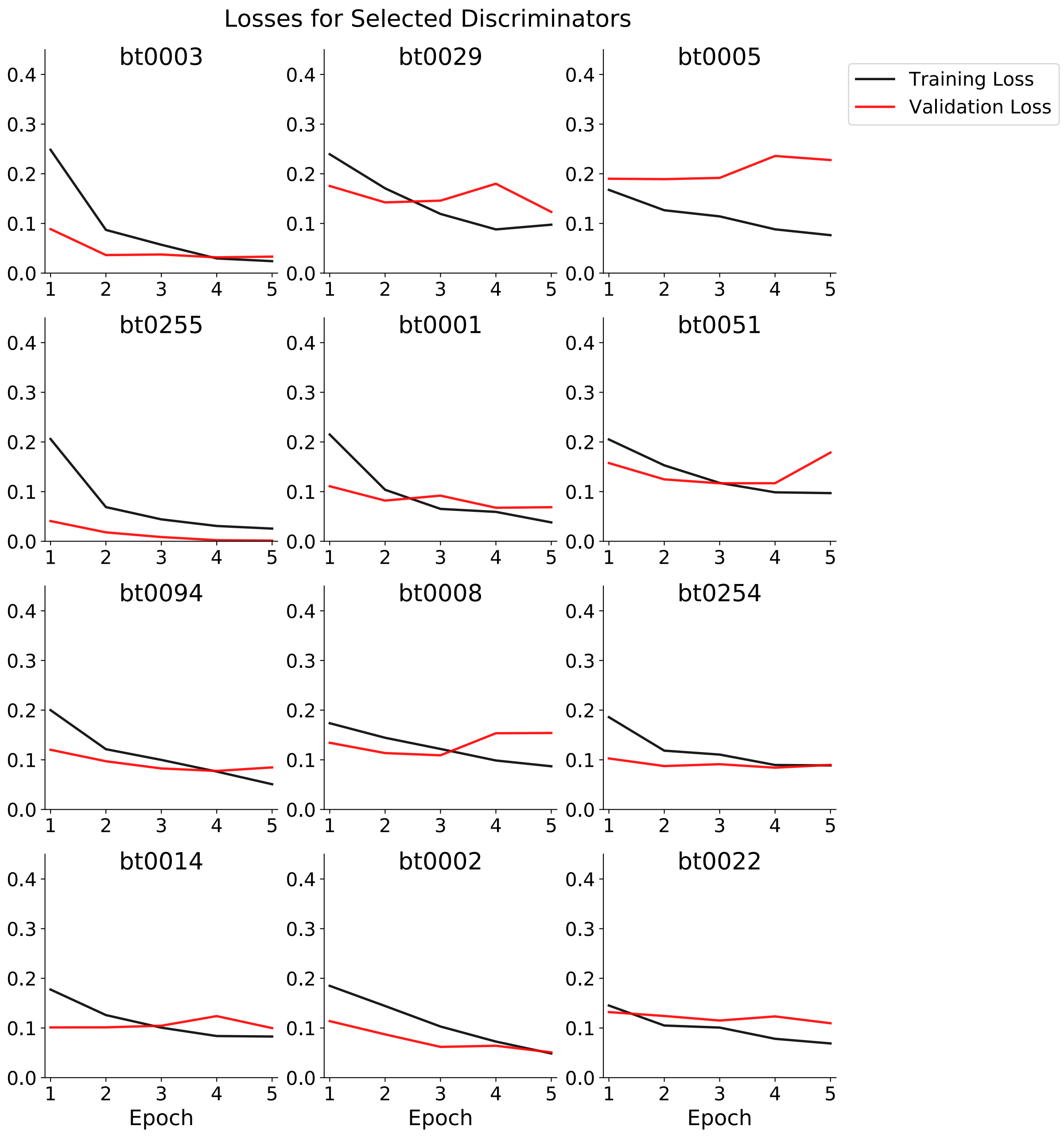


Figure 5.2: Training and validation losses for selected discriminators.

6 Novel Molecule Generation

6.1 Sampling Procedure

In order to sample from the latent space, we need to create a model which can decode a vector to a SMILES. We construct a network similar to the decoder section of the autoencoder, with some adjustments. We require a new input shape, since we will only be feeding this model a single vector. Fortunately, we can transfer the weights of the decoder layers of the autoencoder to the layers of the new model. We also set the LSTM layer in this model to be stateful in order to predict one character at a time.

An illustration of this sampling network can be found in figure 6.1 on page 30

Ideally, the latent space would be normalised to a standard normal distribution $N(0, 1)$, in order to make sampling the neighbouring regions of known points both successful and consistent. Unfortunately the distribution of the latent space in our model does not follow a normal distribution, and we have not modified the model to normalise this layer. However, our experiments showed that adding a 64-dimensional vector randomly sampled from a normal distribution to a known molecule's vector was effective in identifying a diverse range of novel molecules.

We experimented with adding a vector sampled from a normal distribution with a mean of zero and a variety of standard deviations ($N(0, \sigma)$). As the standard deviation, or the distance from the known vector, increased, the success rate of sampling valid molecules decreased. However, at the same time an increase in the distance from the

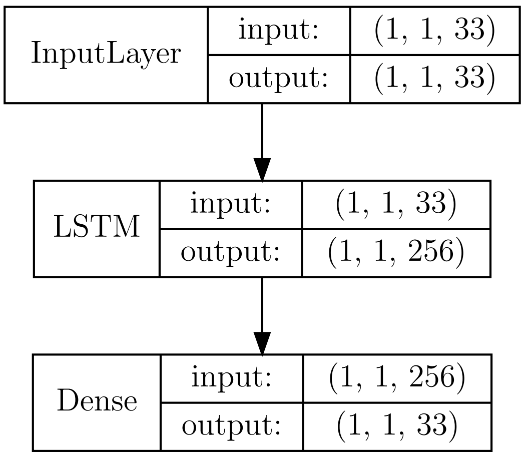


Figure 6.1: Illustration of the sampling model network structure. Note the dimensions - we set the LSTM layer to be stateful in order to sample only one character at a time.

known vector was associated with an increase in the novelty of the sampled molecules.

For each known SMILES in the augmented data set, we took 10 random samples from the neighbourhood, and converted these back to canonical SMILES. We repeated this sampling for standard deviations of 1/3, 2/3, 1, and 3/2. The following table records how many valid SMILES were generated from the 912,990 samples, and then how many remained once duplicates were removed and canonicalisation was applied.

	Standard Deviation (σ) of Sample			
	1/3	2/3	1	3/2
Valid SMILES	860,062	645,308	479,207	352,158
Unique SMILES	137,208	251,468	290,612	278,233
Unique canonical SMILES	27,041	85,877	119,831	139,231
Unique novel canonical SMILES	26,396	85,232	119,193	138,612

We passed each of the sets of unique novel canonical SMILES to the discriminator networks trained for each of the twelve most common biotransformation rules, as measured by the prevalence of molecules triggering each rule in the UM-BBD.

For each molecule which has passed one of the discriminators, we used the Chemical Identifier Resolver (CIR), provided as an API by the Computer-Aided Drug Design (CADD) Group at the (American) National Cancer Institute (NCI), to determine whether an IUPAC (International Union of Pure and Applied Chemistry) name is recorded.

However, due to the intermittent availability of the CIR service, we ultimately switched to using a similar API provided by PubChem [10]. This open database is administered by the (American) National Institutes of Health (NIH) and contains entries for over 236 million compounds.

The presence of a name indicates that the molecule has previously been recorded, and properties of the molecule may already be known. We indicate the percentage of molecules known in PubChem for each biotransformation rule.

Biotransformation	Standard Deviation (σ)				Total	
Rule	1/3	2/3	1	3/2	Unique	PubChem
bt0003	1,844	5,517	7,205	7,921	17,933	16.2%
bt0029	418	1,313	1,949	2,394	4,052	9.6%
bt0255	844	3,296	4,879	5,638	11,811	2.7%
bt0005	100	401	679	982	1,519	17.8%
bt0001	507	1,115	1,292	1,299	2,957	50.9%
bt0002	555	1,060	1,068	853	2,943	18.5%
bt0051	0	0	0	0	0	-
bt0008	20	30	47	39	75	40.0%
bt0094	33	85	104	116	205	8.8%
bt0254	0	0	0	0	0	-
bt0022	9	20	14	14	38	52.6%
bt0014	3	5	3	3	6	66.7%

From this sample, we can clearly see mixed success in generating novel molecules which pass the discriminator. For rules which have had no generated molecules pass the discriminator, this could indicate one or several of the following situations has occurred:

- The discriminator has not been trained successfully;
- Our sampling methodology could be improved;
- The autoencoder has not learnt the features of molecular structures sufficiently.

However, we can also see that we have generated novel molecules which pass one of several biotransformation rules. We can apply a sense check to a sample of these molecules to assess whether we have indeed identified samples which will trigger the given biotransformation rule.

6.2 Samples of Molecules Triggering Biotransformation Rules

In the following sections we illustrate a random sample of molecules generated by our autoencoder which also passed one of the discriminators for the biotransformation rules.

Note that we have not applied any further domain knowledge to assess the reasonableness, feasibility, stability, or any other feature of the molecules listed. Each figure is divided into two sections, the first containing a sample of molecules which exist in PubChem, and the second a sample of molecules which do not.

A full record of all generated molecules which trigger the discriminators is not listed in this paper, due to the length. A complete listing is available from the author on request.

6.2.1 Aldehyde oxidation (bt0003)

This rule is triggered by a large number of our generated molecules (17,933). A small sample of these are illustrated in figure 6.2 on page 33. A cursory examination of this sample shows that all of the molecules are indeed aldehydes, having a C=O double bond, with a hydrogen attached to the carbon (opposed to a ketone, which also has the C=O double bond, but with another hydrocarbon group attached). These molecules therefore all have a feature which suggests they would be good candidates to undergo aldehyde oxidation.

Of the 17,933 molecules, 2,901 (16.2%) have an IUPAC name recorded in PubChem.

6.2.2 Reductive dehalogenation (excluding fluorinated compounds) (bt0029)

This rule is triggered by 4,052 molecules. Of these molecules, 4,014 contain at least one halogen, making them eligible for dehalogenation. The discriminator has learnt that the presence of a halogen indicates that a molecule should trigger this biotransformation rule. A small sample of these are illustrated in figure 6.3 on page 34.

Of the 4,052 molecules, 389 (9.6%) have an IUPAC name recorded.

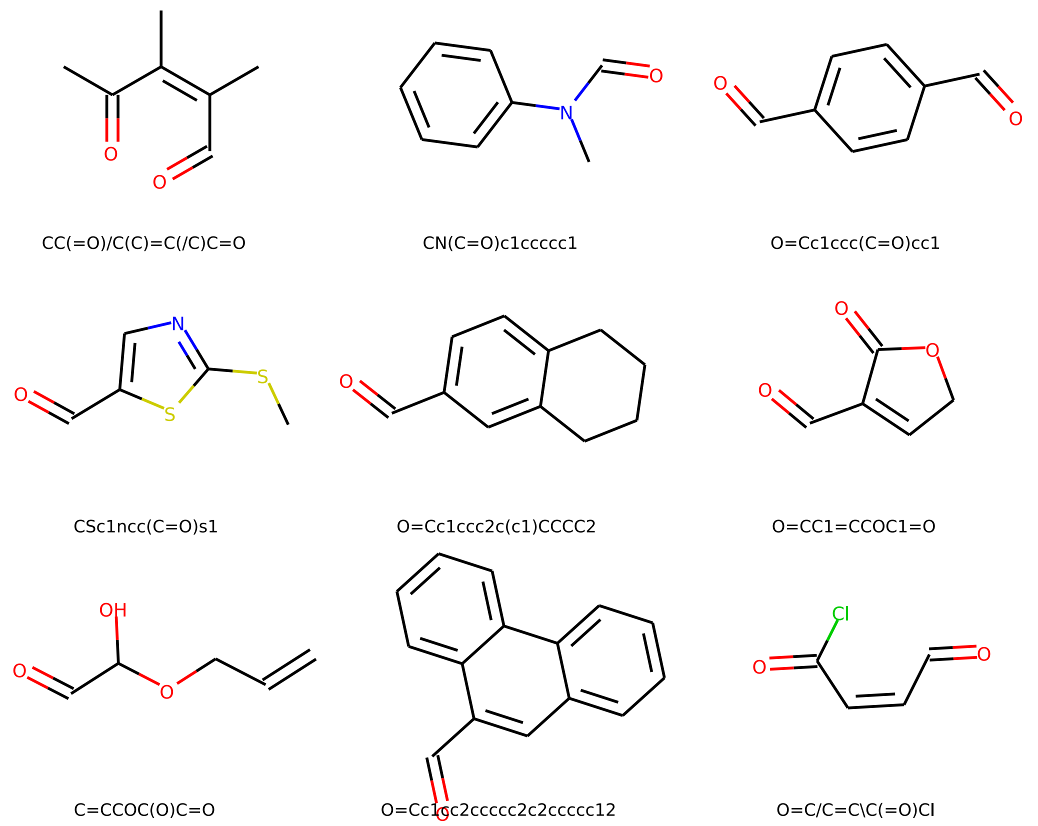
6.2.3 Oxidation of vic-di-H-di-OH to aromatic (bt0255)

This rule represents the aromatisation of a non-aromatic molecule which also has two hydroxy groups attached. An aromatic molecule is a cyclic structure with a ring of resonance bonds, and is particularly stable.

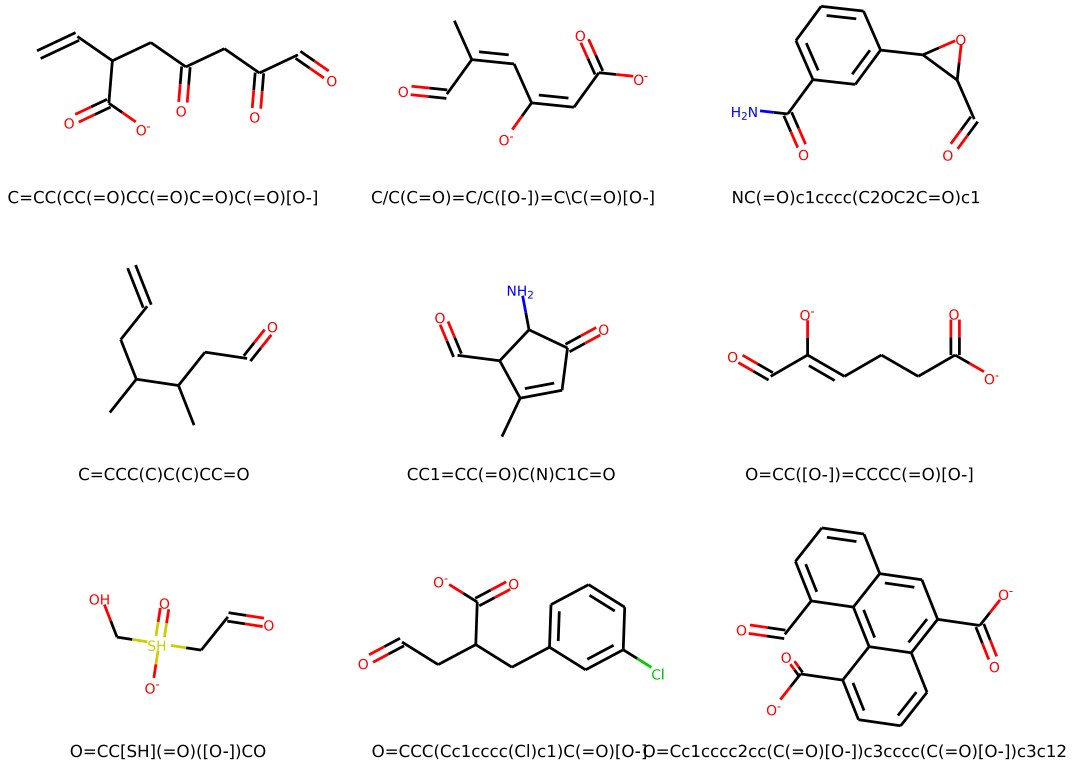
In the simplest example, the rule includes the conversion of 1,2-dihydrobenzene-1,2-diol to benzene-1,2-diol.

By examining the samples in figure 6.4 on page 36, we can see that the discriminator has learnt some key features. For example, most of the molecules contain a non-aromatic cyclic structure, and most also contain two hydroxy groups.

Of the 11,811 molecules, only 318 (2.7%) have an IUPAC name recorded. This is lower than any other rule, and accordingly the unknown molecules identified should be viewed

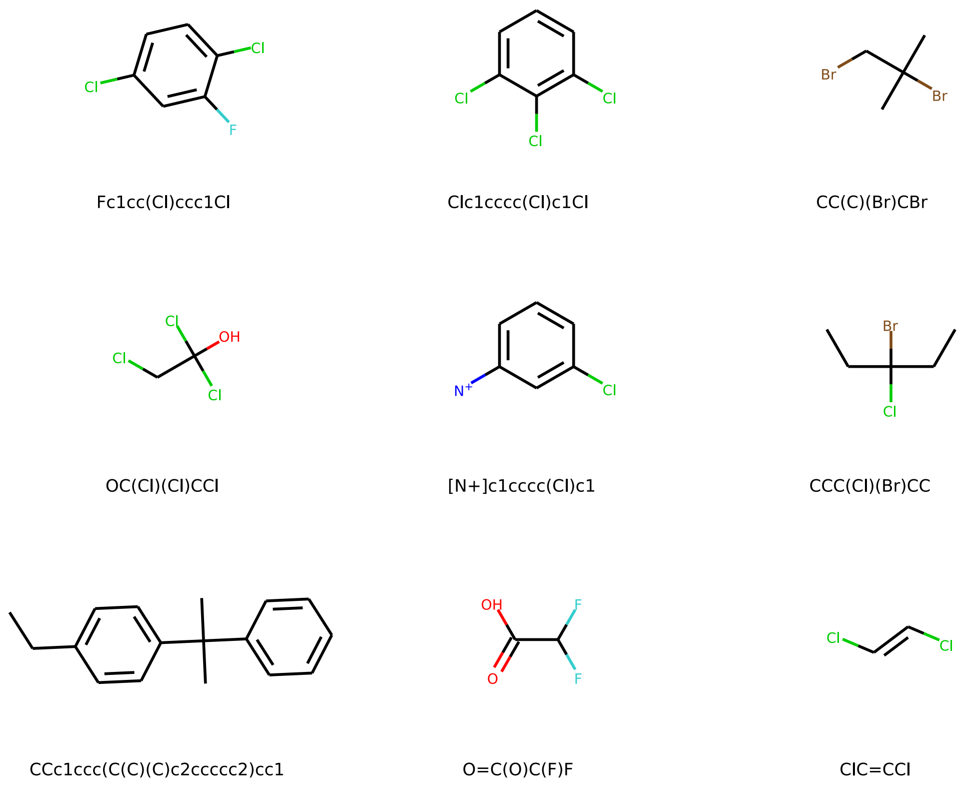


(a) Molecules found in the PubChem database.

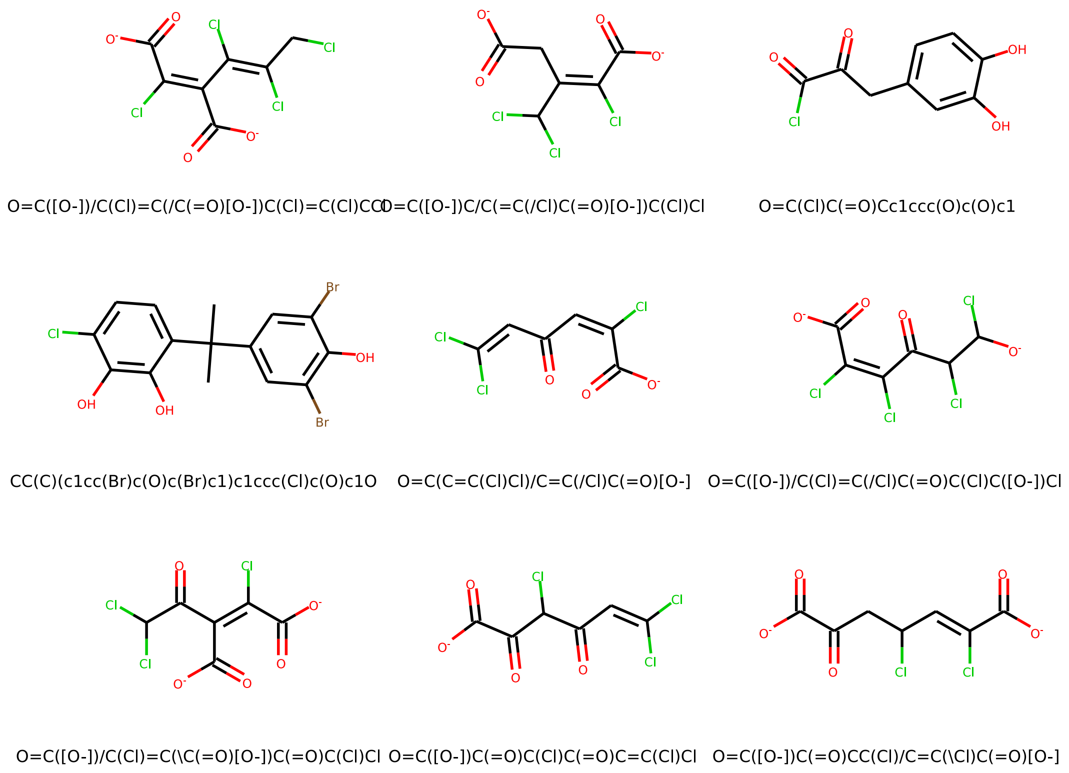

C=CC(CC(=O)CC(=O)C=O)C(=O)[O-]
C/C(C=O)=C/C([O-])=C\C(=O)[O-]
NC(=O)c1cccc(C2OC2C=O)c1
C=CCC(C)C(C)CC=O
CC1=CC(=O)C(N)C1C=O
O=CC([O-])=CCCC(=O)[O-]
O=CC[SH](=O)([O-])CO
O=CCC(Cc1cccc(Cl)c1)C(=O)[O-]
O=Cc1cccc2cc(C(=O)[O-])c3cccc(C(=O)[O-])c3c12

(b) Molecules **not** found in the PubChem database.

Figure 6.2: A sample of the generated molecules which successfully passed the discriminator for the biotransformation rule bt0003.



(a) Molecules found in the PubChem database.


O=C([O-])/C(Cl)=C(/C(=O)[O-])C(Cl)=C(Cl)CCl
O=C([O-])C/C(=C(/Cl)C(=O)[O-])C(Cl)Cl
O=C(Cl)C(=O)Cc1ccc(O)c(O)c1
CC(C)(c1cc(Br)c(O)c(Br)c1)c1ccc(Cl)c(O)c1O
O=C(C=C(Cl)Cl)/C=C(/Cl)C(=O)[O-]
O=C([O-])/C(Cl)=C(/Cl)C(=O)C(Cl)C([O-])Cl
O=C([O-])/C(Cl)=C(\C(=O)[O-])C(=O)C(Cl)Cl
O=C([O-])C(=O)C(Cl)C(=O)C=C(Cl)Cl
O=C([O-])C(=O)CC(Cl)/C=C(\Cl)C(=O)[O-]

(b) Molecules **not** found in the PubChem database.

Figure 6.3: A sample of the generated molecules which successfully passed the discriminator for the biotransformation rule bt0029.

with appropriate scepticism. While the sample of molecules not found in PubChem do appear to have the features required by this rule, several also have extremely unusual structures.

6.2.4 Aromatic ring dioxygenation (bt0005)

This rule represents the dioxygenation of an aromatic ring. Dioxygenation of aromatic compounds is an important initial reaction in the bacterial degradation process [18].

The discriminator has learnt to identify aromatic rings, as seen in figure 6.5 on page 37 and indicated by the lower case SMILES. However, many of those generated are not known in PubChem, and have visibly unusual structures.

Of the 1,519 molecules, 271 (17.8%) have an IUPAC name recorded.

6.2.5 Primary alcohol oxidation (bt0001)

This rule is triggered by 2,957 molecules. All the identified molecules seem to have at least one alcohol group, suggesting that they are good candidates for alcohol oxidation. Furthermore, the discriminator appears to have learnt to identify primary alcohols (those with the hydroxyl functional group attached to a carbon atom only attached to one other carbon atom).

Of the 2,957 molecules, 1,505 (50.9%) have an IUPAC name recorded. A sample of these is presented in figure 6.6 on page 38.

6.2.6 Secondary alcohol oxidation (bt0002)

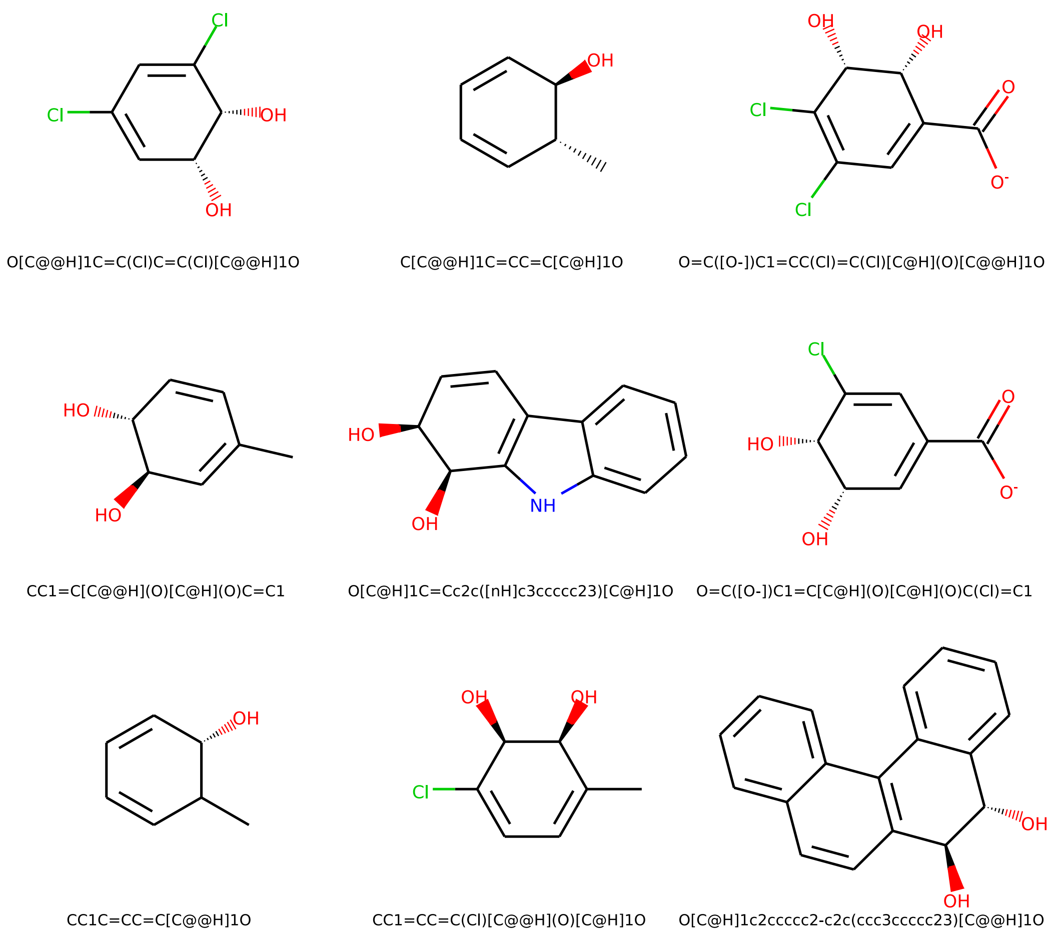
The discriminator for this rule seems to have learnt how to identify secondary alcohols, but chiefly those with a ring structure.

Of the 2,943 molecules, 544 (18.5%) have an IUPAC name recorded. A sample of these is presented in figure 6.7 on page 39.

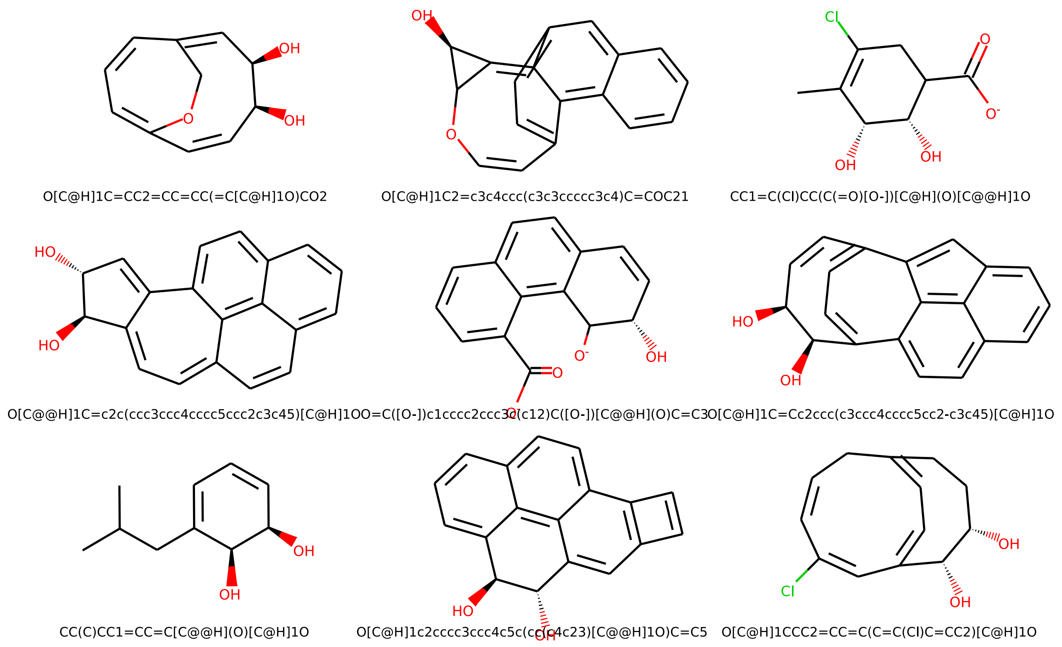
6.2.7 Aromatic vic-diol ring cleavage (extradiol ring cleavage of vic-dihydroxybenzenoids) (bt0008)

This rule represents the cleavage (or breaking) of a ring in a single or multi-ring aromatic compound. Specifically, it requires a benzene-like molecule with two hydroxy groups. The discriminator appears to have learnt this rule well.

Of the 75 molecules, 30 (40.0%) have an IUPAC name recorded. A sample of these is presented in figure 6.8 on page 40.

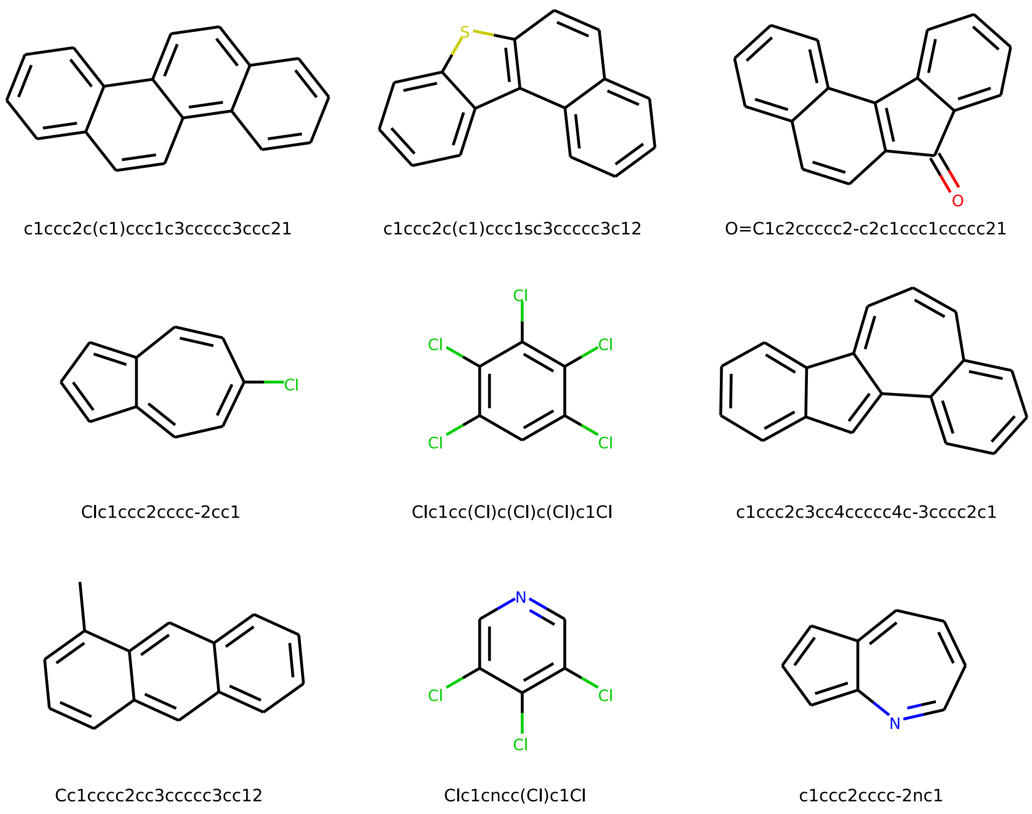

O[C@@H]1C=C(Cl)C=C(Cl)[C@@H]1O
C[C@@H]1C=CC=C[C@H]1O
O=C([O-])C1=CC(Cl)=C(Cl)[C@H](O)[C@@H]1O
CC1=C[C@@H](O)[C@H](O)C=C1
O[C@H]1C=Cc2c([nH]c3ccccc23)[C@H]1O
O=C([O-])C1=C[C@H](O)[C@H](O)C(Cl)=C1
CC1C=CC=C[C@@H]1O
CC1=CC=C(Cl)[C@@H](O)[C@H]1O
O[C@H]1c2ccccc2-c2c(ccc3ccccc23)[C@@H]1O

(a) Molecules found in the PubChem database.

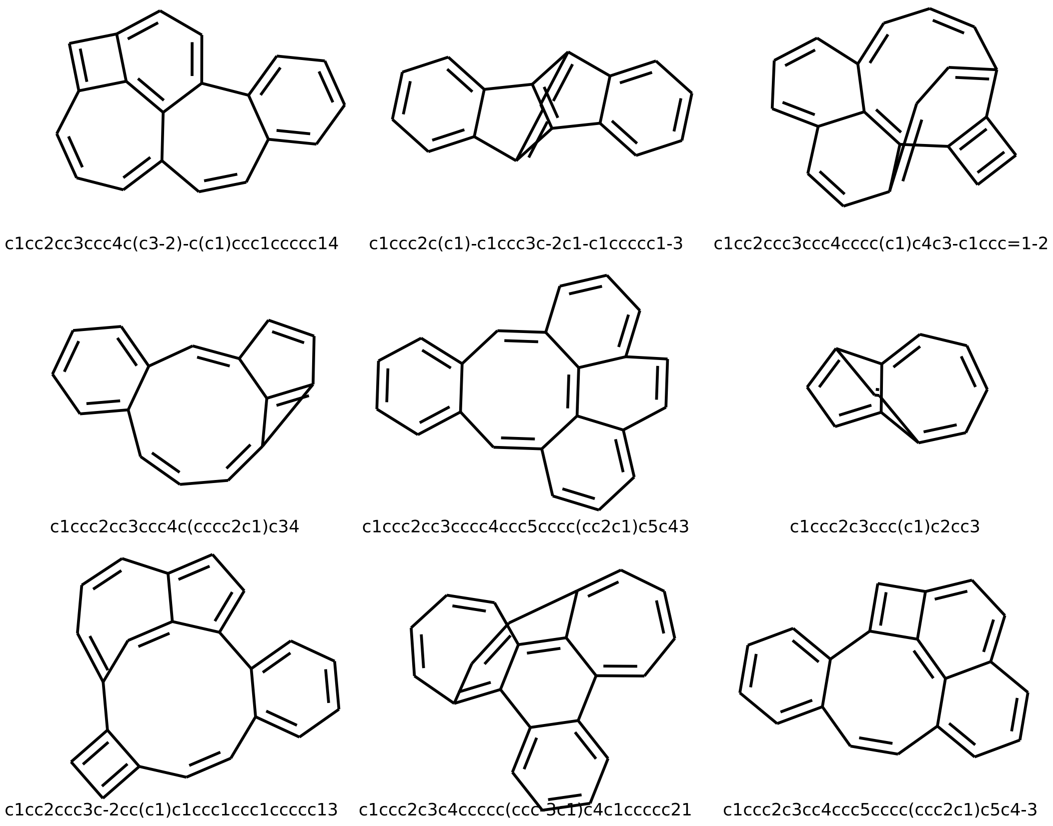

O[C@H]1C=CC2=CC=CC(=C[C@H]1O)CO2
O[C@H]1C2=c3c4ccc(c3c3ccccc3c4)C=COC21
CC1=C(Cl)CC(C(=O)[O-])[C@H](O)[C@@H]1O
O[C@@H]1C=c2c(ccc3ccc4cccc5ccc2c3c45)[C@H]1O
O=C([O-])c1cccc2ccc3c(c12)C([O-])[C@@H](O)C=C3
O[C@H]1C=Cc2ccc(c3ccc4cccc5cc2-c3c45)[C@H]1O
CC(C)CC1=CC=C[C@@H](O)[C@H]1O
O[C@@H]1c2cccc3ccc4c5c(cc(c4c23)[C@@H]1O)C=C5
O[C@H]1CCC2=CC=C(C=C(Cl)C=CC2)[C@H]1O

(b) Molecules **not** found in the PubChem database.

Figure 6.4: A sample of the generated molecules which successfully passed the discriminator for the biotransformation rule bt0255.

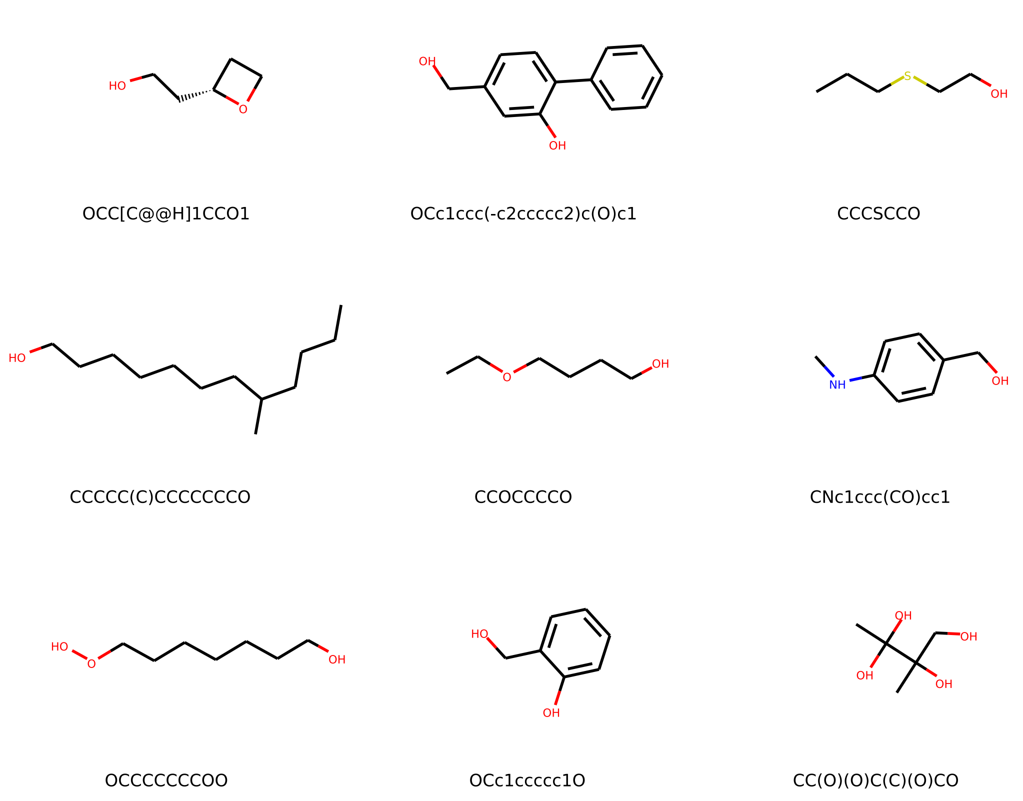

c1ccc2c(c1)ccc1c3ccccc3ccc21
c1ccc2c(c1)ccc1sc3ccccc3c12
O=C1c2ccccc2-c2c1ccc1ccccc21
Clc1ccc2cccc-2cc1
Clc1cc(Cl)c(Cl)c(Cl)c1Cl
c1ccc2c3cc4ccccc4c-3cccc2c1
Cc1cccc2cc3ccccc3cc12
Clc1cncc(Cl)c1Cl
c1ccc2cccc-2nc1

(a) Molecules found in the PubChem database.

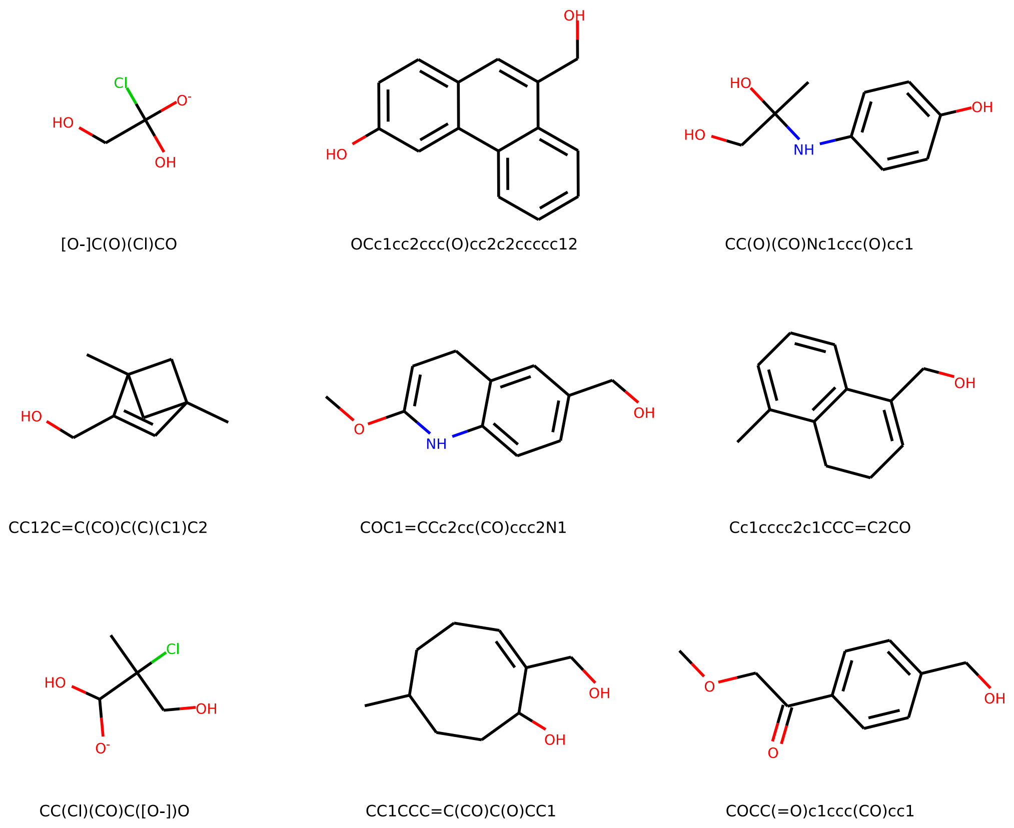

c1cc2cc3ccc4c(c3-2)-c(c1)ccc1ccccc14
c1ccc2c(c1)-c1ccc3c-2c1-c1ccccc1-3
c1cc2ccc3ccc4cccc(c1)c4c3-c1ccc=1-2
c1ccc2cc3ccc4c(cccc2c1)c34
c1ccc2cc3cccc4ccc5cccc(cc2c1)c5c43
c1ccc2c3ccc(c1)c2cc3
c1cc2ccc3c-2cc(c1)c1ccc1ccc1ccccc13
c1ccc2c3c4ccccc(ccc3c1)c4c1ccccc21
c1ccc2c3cc4ccc5cccc(ccc2c1)c5c4-3

(b) Molecules **not** found in the PubChem database.

Figure 6.5: A sample of the generated molecules which successfully passed the discriminator for the biotransformation rule bt0005.

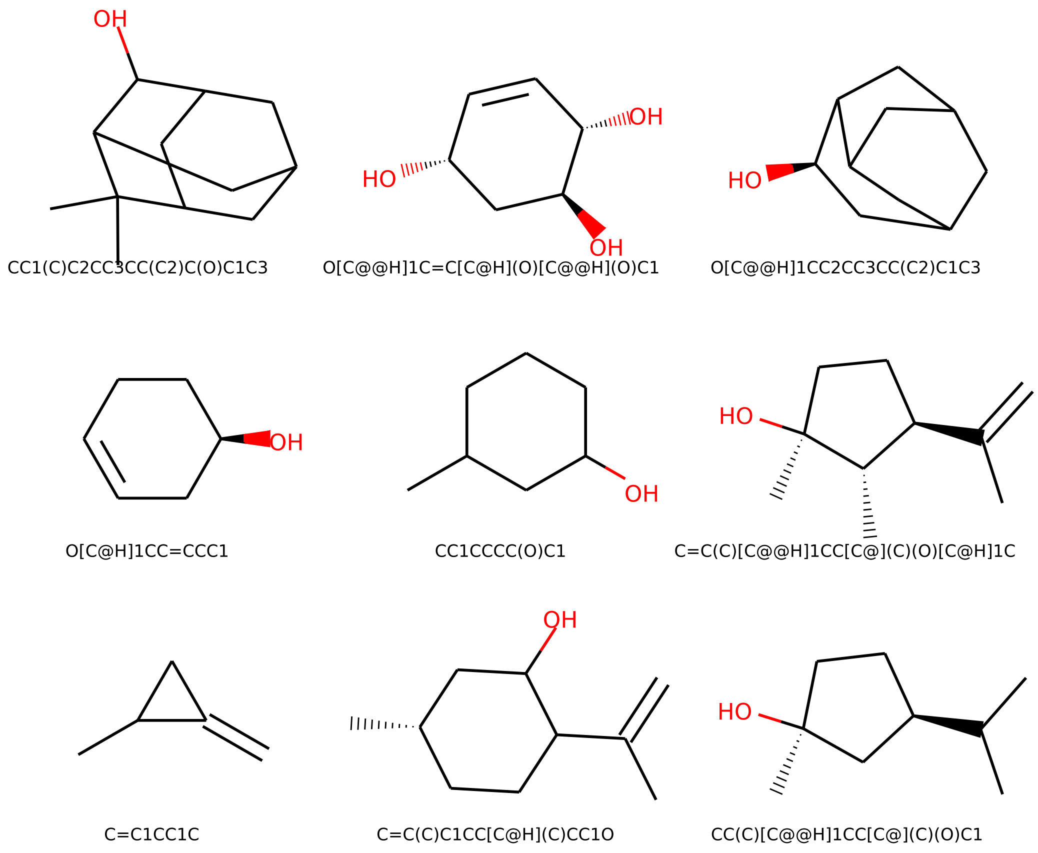


(a) Molecules found in the PubChem database.

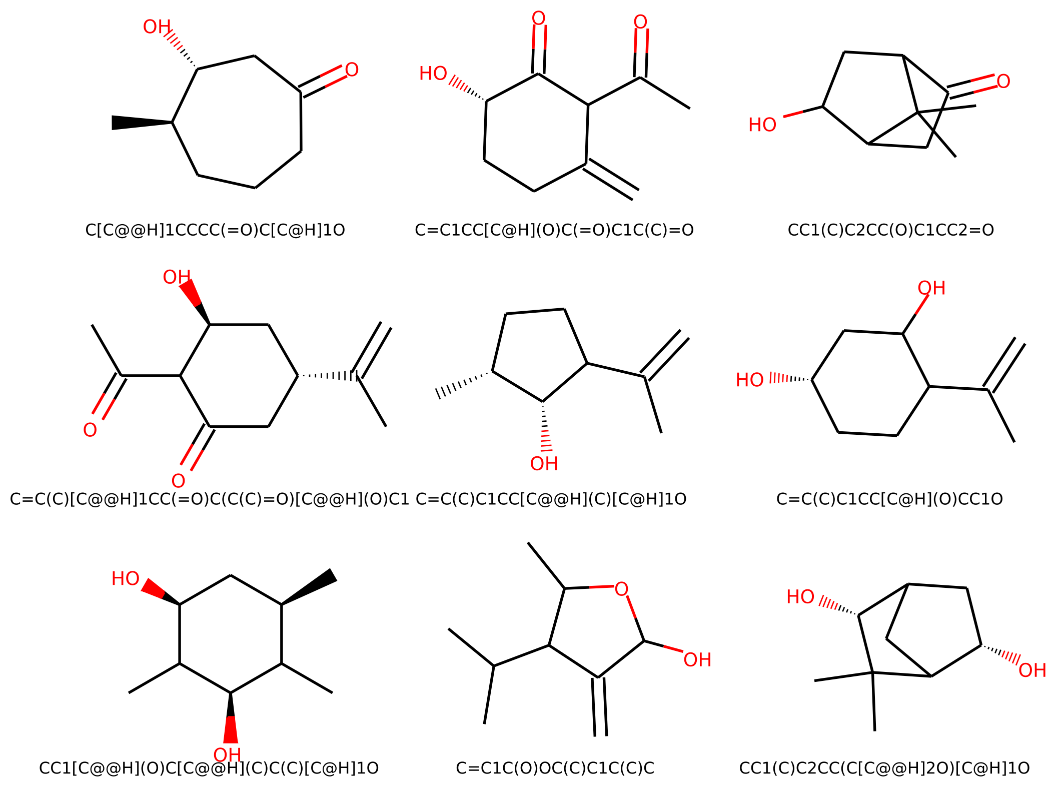


(b) Molecules **not** found in the PubChem database.

Figure 6.6: A sample of the generated molecules which successfully passed the discriminator for the biotransformation rule bt0001.

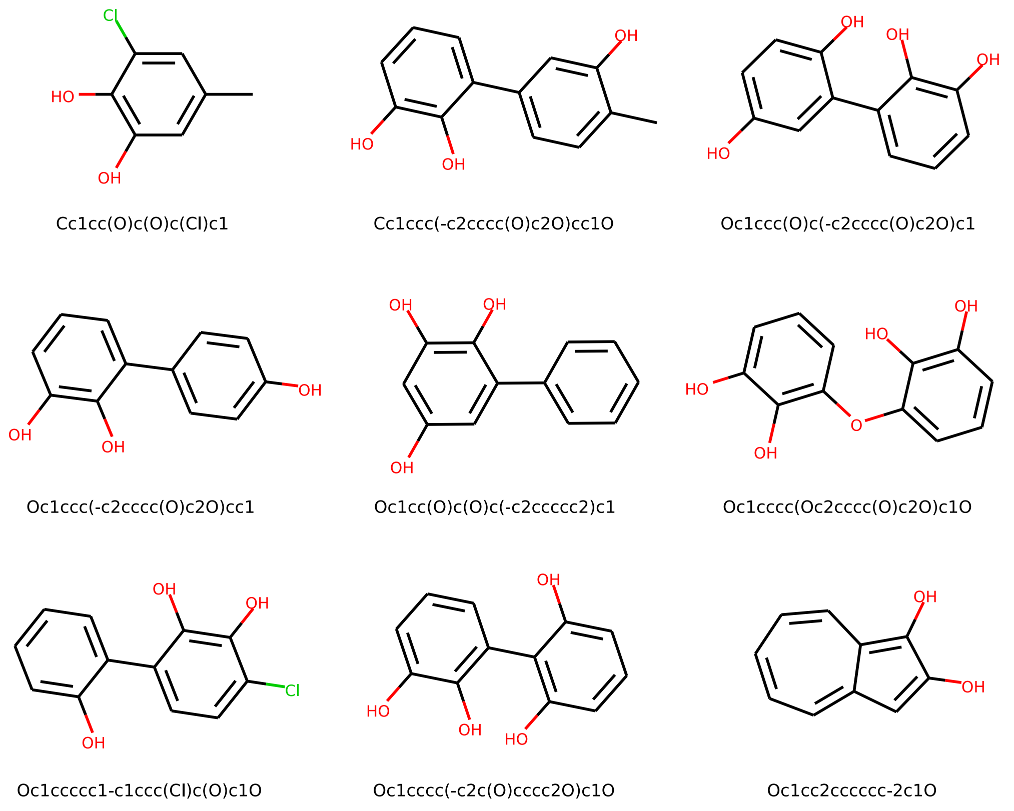

CC1(C)C2CC3CC(C2)C(O)C1C3
O[C@@H]1C=C[C@H](O)[C@@H](O)C1
O[C@@H]1CC2CC3CC(C2)C1C3
O[C@H]1CC=CCC1
CC1CCCC(O)C1
C=C(C)[C@@H]1CC[C@](C)(O)[C@H]1C
C=C1CC1C
C=C(C)C1CC[C@H](C)CC1O
CC(C)[C@@H]1CC[C@](C)(O)C1

(a) Molecules found in the PubChem database.

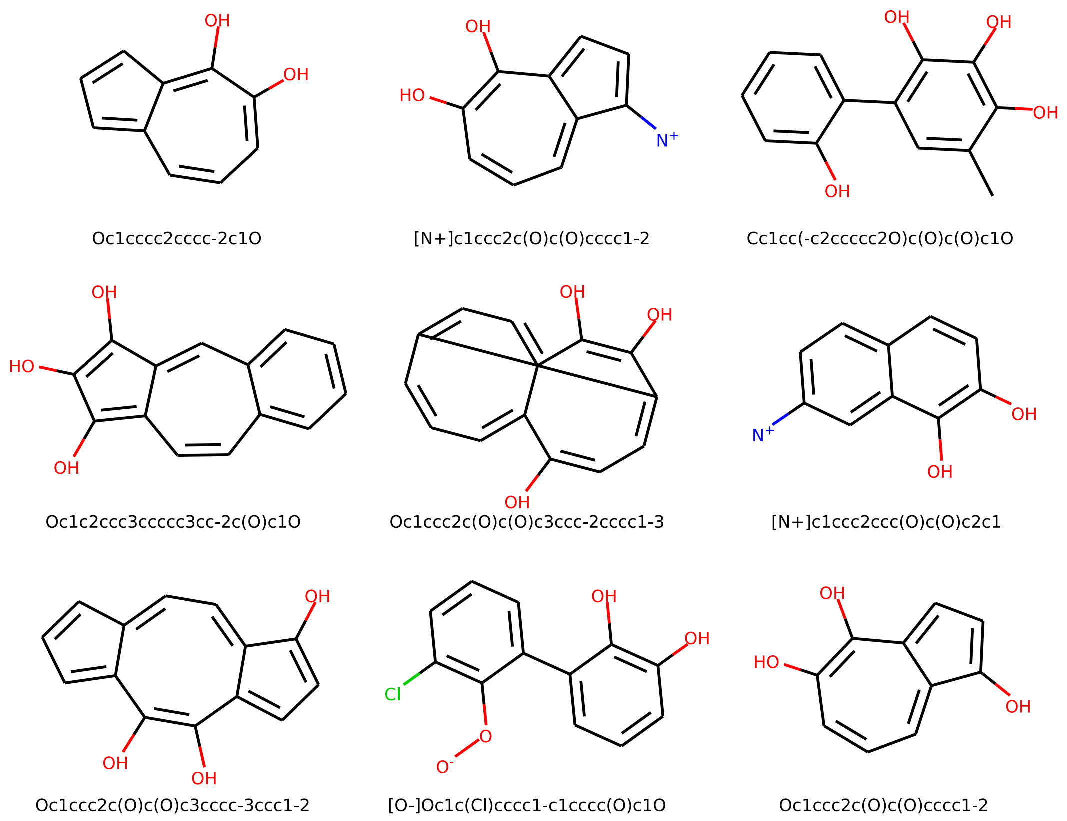

C[C@@H]1CCCC(=O)C[C@H]1O
C=C1CC[C@H](O)C(=O)C1C(C)=O
CC1(C)C2CC(O)C1CC2=O
C=C(C)[C@@H]1CC(=O)C(C(C)=O)[C@@H](O)C1
C=C(C)C1CC[C@@H](C)[C@H]1O
C=C(C)C1CC[C@H](O)CC1O
CC1[C@@H](O)C[C@@H](C)C(C)[C@H]1O
C=C1C(O)OC(C)C1C(C)C
CC1(C)C2CC(C[C@@H]2O)[C@H]1O

(b) Molecules **not** found in the PubChem database.

Figure 6.7: A sample of the generated molecules which successfully passed the discriminator for the biotransformation rule bt0002.


Cc1cc(O)c(O)c(Cl)c1
Cc1ccc(-c2cccc(O)c2O)cc1O
Oc1ccc(O)c(-c2cccc(O)c2O)c1
Oc1ccc(-c2cccc(O)c2O)cc1
Oc1cc(O)c(O)c(-c2ccccc2)c1
Oc1cccc(Oc2cccc(O)c2O)c1O
Oc1ccccc1-c1ccc(Cl)c(O)c1O
Oc1cccc(-c2c(O)cccc2O)c1O
Oc1cc2ccccccc-2c1O

(a) Molecules found in the PubChem database.


Oc1cccc2cccc-2c1O
[N+]c1ccc2c(O)c(O)cccc1-2
Cc1cc(-c2ccccc2O)c(O)c(O)c1O
Oc1c2ccc3ccccc3cc-2c(O)c1O
Oc1ccc2c(O)c(O)c3ccc-2cccc1-3
[N+]c1ccc2ccc(O)c(O)c2c1
Oc1ccc2c(O)c(O)c3cccc-3ccc1-2
[O-]Oc1c(Cl)cccc1-c1cccc(O)c1O
Oc1ccc2c(O)c(O)cccc1-2

(b) Molecules **not** found in the PubChem database.

Figure 6.8: A sample of the generated molecules which successfully passed the discriminator for the biotransformation rule bt0008.

6.2.8 Coenzyme A-thioester formation (bt0094)

A prerequisite for the formation of a thioester is a carboxylic acid. The discriminator has in many cases learnt the key feature of a carboxylic acid, the -COOH group.

Of the 205 molecules, 18 (8.8%) have an IUPAC name recorded. A sample of these is presented in figure 6.9 on page 42.

6.2.9 Hydrolytic dehalogenation of aliphatic halomethyl and dihalomethyl derivatives (bt0022)

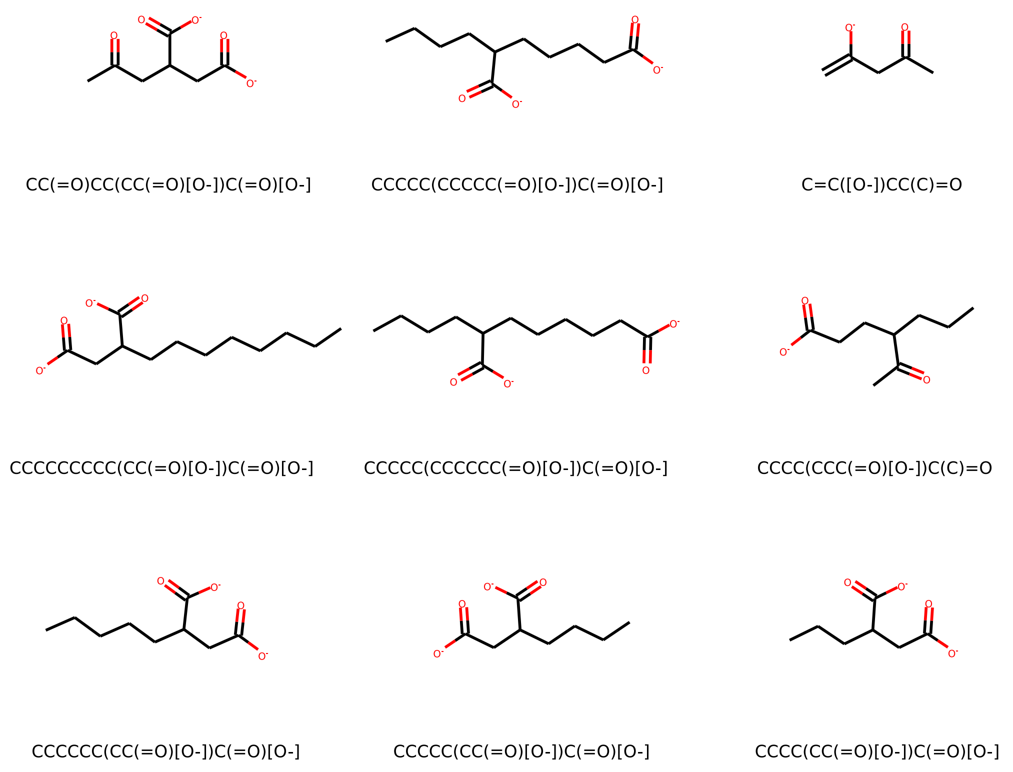
Similar to the discriminator for rule bt0029, the discriminator for this rule has learnt to identify molecules with multiple halogens. Given that this biotransformation rule indicates a removal of a halogen, these molecules seem to be good candidates.

Of the 38 molecules, 20 (52.6%) have an IUPAC name recorded. A sample of these is presented in figure 6.10 on page 43.

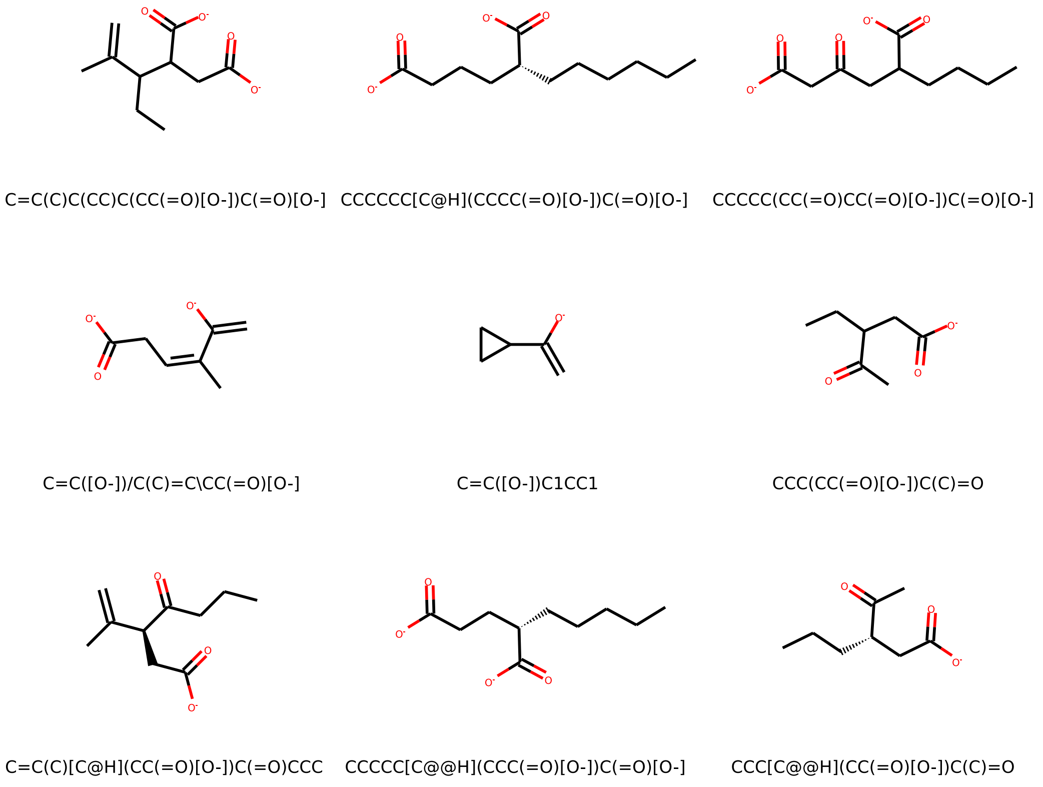
6.2.10 Creation of a vic-dihydroxyl aromatic (bt0014)

Only a small number of molecules were triggered by the discriminator for this biotransformation rule. The rule represents the hydroxylation of an aromatic to create an aromatic ring with two hydroxy groups. The molecules generated are aromatic, and generally have already at least one hydroxy group.

Of the 6 molecules, 4 (66.7%) have an IUPAC name recorded. A sample of these is presented in figure 6.11 on page 44.

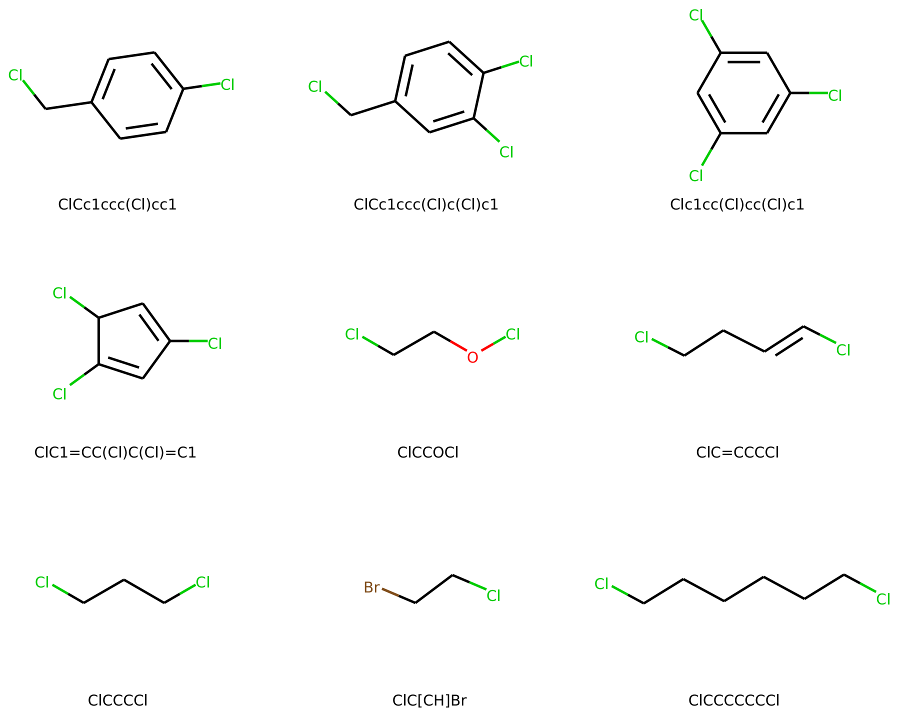

CC(=O)CC(CC(=O)[O-])C(=O)[O-]
CCCCC(CCCCC(=O)[O-])C(=O)[O-]
C=C([O-])CC(C)=O
CCCCCCCCC(CC(=O)[O-])C(=O)[O-]
CCCCC(CCCCCC(=O)[O-])C(=O)[O-]
CCCC(CCC(=O)[O-])C(C)=O
CCCCCC(CC(=O)[O-])C(=O)[O-]
CCCCC(CC(=O)[O-])C(=O)[O-]
CCCC(CC(=O)[O-])C(=O)[O-]

(a) Molecules found in the PubChem database.

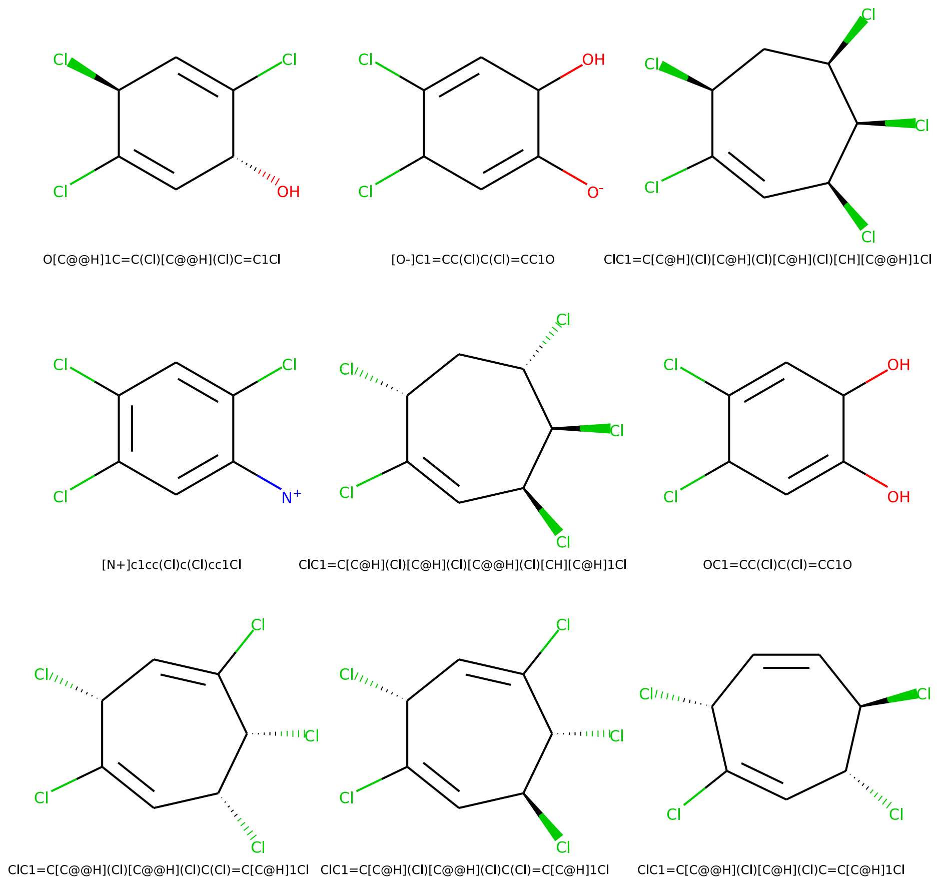

C=C(C)C(CC)C(CC(=O)[O-])C(=O)[O-]
CCCCCC[C@H](CCCC(=O)[O-])C(=O)[O-]
CCCCC(CC(=O)CC(=O)[O-])C(=O)[O-]
C=C([O-])/C(C)=C\CC(=O)[O-]
C=C([O-])C1CC1
CCC(CC(=O)[O-])C(C)=O
C=C(C)[C@H](CC(=O)[O-])C(=O)CCC
CCCCC[C@@H](CCC(=O)[O-])C(=O)[O-]
CCC[C@@H](CC(=O)[O-])C(C)=O

(b) Molecules **not** found in the PubChem database.

Figure 6.9: A sample of the generated molecules which successfully passed the discriminator for the biotransformation rule bt0094.

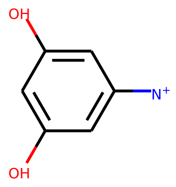


(a) Molecules found in the PubChem database.

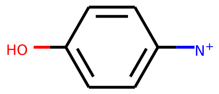

O[C@@H]1C=C(Cl)[C@@H](Cl)C=C1Cl
[O-]C1=CC(Cl)C(Cl)=CC1O
ClC1=C[C@H](Cl)[C@H](Cl)[C@H](Cl)[CH][C@@H]1Cl
[N+]c1cc(Cl)c(Cl)cc1Cl
ClC1=C[C@H](Cl)[C@H](Cl)[C@@H](Cl)[CH][C@H]1Cl
OC1=CC(Cl)C(Cl)=CC1O
ClC1=C[C@@H](Cl)[C@@H](Cl)C(Cl)=C[C@H]1Cl
ClC1=C[C@H](Cl)[C@@H](Cl)C(Cl)=C[C@H]1Cl
ClC1=C[C@@H](Cl)[C@H](Cl)C=C[C@H]1Cl

(b) Molecules **not** found in the PubChem database.

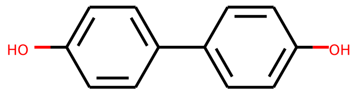
Figure 6.10: A sample of the generated molecules which successfully passed the discriminator for the biotransformation rule bt0022.



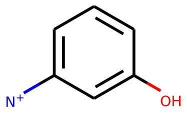
[N+]c1cc(O)cc(O)c1



[N+]c1ccc(O)cc1

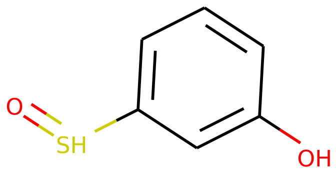


Oc1ccc(-c2ccc(O)cc2)cc1

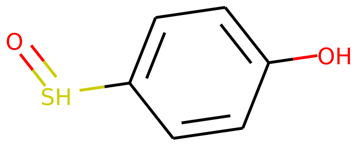


[N+]c1cccc(O)c1

(a) Molecules found in the PubChem database.



O=[SH]c1cccc(O)c1



O=[SH]c1ccc(O)cc1

(b) Molecules **not** found in the PubChem database.

Figure 6.11: A sample of the generated molecules which successfully passed the discriminator for the biotransformation rule bt0014.

7 Conclusion

7.1 Achievements and Significance

Through leveraging sequence-to-sequence learning methods from the field of natural language processing, we have successfully constructed an autoencoder which encodes SMILES strings in a variety of low-dimensional spaces. Our best result shows a high level of reconstruction of the set of hold out molecules (99.9% of characters, and 94.9% of complete SMILES). This differs from previous work (e.g. "mol2vec") in two key ways:

- We were able to achieve this with an extremely small data set (646 molecules vs. ~20,000,000). This demonstrates that large volumes of data are not necessarily required for encoding representations of chemical compounds into a latent space.
- We have achieved this solely using SMILES, and no other representation of the molecule, such as ECFP fingerprints.

Interrogating the latent space has shown that molecules triggered by different biotransformation rules reside for the most part in separate regions, indicating that our autoencoder has learnt key features of different groups of molecules.

Furthermore, we have successfully created networks which appear to accurately identify whether generated molecules would be triggered by a given biotransformation rule. Depending on the rule, between 2.7% and 66.7% of these molecules already exist in the

PubChem database. This indicates that our generator is moderately good at generating molecules with desired properties, and that is likely that at least some of those molecules which do not exist in PubChem are candidates for further study.

7.2 Future Work

We have identified several possible areas of interesting future research.

7.2.1 Use of a Generative Adversarial Network

In conducting these experiments, we were not able to construct a full Generative Adversarial Network (GAN). A key challenge was the different representations used of molecules in our autoencoder and our discriminator networks.

We hypothesise that the use of a GAN may enable the generation of molecules which are both more likely to exist, and more likely to trigger a given discriminator.

7.2.2 Include Molecule Name Data

The key principle behind a GAN is that the discriminator can learn to identify generated samples from real samples. If we consider that a real SMILES is one that has a valid representation in the PubChem database, then we could possibly train the discriminator with this information, and hence proceed with iterative GAN training.

7.2.3 Augment Data Set for Training Discriminator

Data augmentation may improve the discriminators used in this research. It may be possible to take the biotransformation rules and match them to a database of known molecules using a rule-based approach.

7.2.4 Normalisation of Latent Space

The data in our latent (hidden) space is not normally distributed, and therefore encoded information is not evenly distributed through the space. Most of the variation is likely to be concentrated in small regions of the latent space. This means that sampling around known data points using normally distributed noise cannot give consistently good outputs, and sampling randomly in the space is impossible.

Research suggests that if we could apply some kind of transformation to the latent space during model training, this would address the issue [21]. The method would likely involve the Kullback-Leibler Divergence (KL-Divergence), which is a measure of how one probability distribution differs from a baseline distribution.

7.2.5 More Data

Any source of additional data would significantly help the robustness of our models, particularly the discriminators. These can easily be retrained with new data sets.

7.2.6 Analysis of Generated Molecules

Our discriminators have identified generated molecules which may trigger certain biotransformation rules, but are unknown in the PubChem database. An expert in the relevant sub-field of chemistry may be able to review our set of unknown generated molecules and identify which are likely to be stable and have desired properties through the application of domain knowledge or specific rules.

8 Appendix

8.1 Autoencoder Training Statistics

The following tables record the training loss, validation loss, and reconstruction statistics for training the autoencoder using different parameters, as described in section 4.5 *Model Training.*

Latent Dimensions	LSTM Units	Original Data 2	4	8	16
2	Training loss [a]	0.915	0.630	0.469	0.413
	Validation loss [b]	0.907	0.605	0.450	0.402
	SMILES reconstruction [c]	0.0%	0.0%	0.0%	3.8%
	Character reconstruction [d]	72.0%	82.8%	86.9%	87.3%
4	Training loss [a]	0.862	0.643	0.499	0.381
	Validation loss [b]	0.861	0.608	0.481	0.379
	SMILES reconstruction [c]	0.0%	0.0%	0.0%	0.0%
	Character reconstruction [d]	71.8%	80.6%	84.5%	87.0%
8	Training loss [a]	0.940	0.688	0.467	0.398
	Validation loss [b]	0.940	0.656	0.451	0.386
	SMILES reconstruction [c]	0.0%	0.0%	7.7%	0.0%
	Character reconstruction [d]	65.1%	78.7%	87.5%	86.6%
16	Training loss [a]	0.873	0.651	0.511	0.388
	Validation loss [b]	0.877	0.625	0.486	0.377
	SMILES reconstruction [c]	0.0%	0.0%	0.0%	0.0%
	Character reconstruction [d]	71.7%	80.0%	84.5%	87.7%
32	Training loss [a]	0.863	0.697	0.514	0.409
	Validation loss [b]	0.846	0.673	0.495	0.396
	SMILES reconstruction [c]	0.0%	0.0%	0.0%	0.0%
	Character reconstruction [d]	72.7%	78.9%	85.0%	87.4%
64	Training loss [a]	0.854	0.660	0.507	0.392
	Validation loss [b]	0.839	0.637	0.479	0.390
	SMILES reconstruction [c]	0.0%	0.0%	0.0%	0.0%
	Character reconstruction [d]	71.4%	80.8%	84.9%	86.5%

[a] The loss evaluated against the training data set after 100 epochs.
[b] The loss evaluated against the validation data set after 100 epochs.
[c] The percentage of SMILES in the holdout data set which were completely reconstructed.
[d] The percentage of characters in SMILES in the holdout data set which were successfully reconstructed.

Latent Dimensions	LSTM Units	Original Data 32	64	128	256
2	Training loss [a]	0.304	0.223	0.181	0.178
	Validation loss [b]	0.345	0.348	0.336	0.331
	SMILES reconstruction [c]	0.0%	0.0%	0.0%	0.0%
	Character reconstruction [d]	88.9%	89.1%	88.8%	89.2%
4	Training loss [a]	0.274	0.155	0.170	0.140
	Validation loss [b]	0.325	**0.294**	0.334	0.333
	SMILES reconstruction [c]	0.0%	3.8%	0.0%	0.0%
	Character reconstruction [d]	89.9%	91.0%	88.7%	90.5%
8	Training loss [a]	0.262	0.192	0.144	0.087
	Validation loss [b]	0.301	0.300	0.314	0.311
	SMILES reconstruction [c]	0.0%	0.0%	0.0%	3.8%
	Character reconstruction [d]	90.4%	90.9%	90.1%	91.5%
16	Training loss [a]	0.331	0.153	**0.129**	0.131
	Validation loss [b]	0.363	0.296	0.303	0.335
	SMILES reconstruction [c]	0.0%	0.0%	0.0%	0.0%
	Character reconstruction [d]	88.9%	90.7%	89.6%	89.6%
32	Training loss [a]	0.242	0.184	0.190	0.180
	Validation loss [b]	0.307	0.304	0.338	0.335
	SMILES reconstruction [c]	**11.5%**	0.0%	0.0%	0.0%
	Character reconstruction [d]	**91.8%**	90.1%	88.9%	88.5%
64	Training loss [a]	0.275	0.230	0.208	0.134
	Validation loss [b]	0.362	0.327	0.339	0.316
	SMILES reconstruction [c]	0.0%	0.0%	0.0%	0.0%
	Character reconstruction [d]	89.5%	88.6%	88.9%	89.6%

[a] The loss evaluated against the training data set after 100 epochs.
[b] The loss evaluated against the validation data set after 100 epochs.
[c] The percentage of SMILES in the holdout data set which were completely reconstructed.
[d] The percentage of characters in SMILES in the holdout data set which were successfully reconstructed.

Latent Dimensions	LSTM Units	Augmented Data 2	4	8	16
2	Training loss [a]	1.060	0.818	0.688	0.477
	Validation loss [b]	1.060	0.819	0.689	0.477
	SMILES reconstruction [c]	0.0%	0.0%	0.0%	0.0%
	Character reconstruction [d]	66.9%	72.1%	76.0%	82.1%
4	Training loss [a]	1.004	0.833	0.633	0.464
	Validation loss [b]	1.005	0.834	0.632	0.461
	SMILES reconstruction [c]	0.0%	0.0%	0.0%	0.0%
	Character reconstruction [d]	69.0%	73.2%	78.3%	83.4%
8	Training loss [a]	1.023	0.822	0.610	0.465
	Validation loss [b]	1.024	0.822	0.612	0.460
	SMILES reconstruction [c]	0.0%	0.0%	0.0%	0.1%
	Character reconstruction [d]	67.5%	73.2%	79.0%	83.4%
16	Training loss [a]	1.107	0.773	0.618	0.416
	Validation loss [b]	1.106	0.769	0.618	0.414
	SMILES reconstruction [c]	0.0%	0.0%	0.1%	0.1%
	Character reconstruction [d]	66.6%	75.2%	79.0%	85.3%
32	Training loss [a]	1.136	0.803	0.684	0.416
	Validation loss [b]	1.140	0.804	0.684	0.420
	SMILES reconstruction [c]	0.0%	0.0%	0.0%	0.1%
	Character reconstruction [d]	61.1%	74.4%	76.3%	85.0%
64	Training loss [a]	1.055	0.826	0.660	0.422
	Validation loss [b]	1.054	0.827	0.660	0.420
	SMILES reconstruction [c]	0.0%	0.0%	0.0%	0.1%
	Character reconstruction [d]	66.4%	73.4%	76.9%	85.3%

[a] The loss evaluated against the training data set after 100 epochs.

[b] The loss evaluated against the validation data set after 100 epochs.

[c] The percentage of SMILES in the holdout data set which were completely reconstructed.

[d] The percentage of characters in SMILES in the holdout data set which were successfully reconstructed.

Latent Dimensions	LSTM Units	Augmented Data 32	64	128	256
2	Training loss [a]	0.331	0.190	0.243	0.232
	Validation loss [b]	0.346	0.198	0.260	0.252
	SMILES reconstruction [c]	0.0%	0.9%	0.0%	0.0%
	Character reconstruction [d]	86.9%	92.2%	88.0%	87.9%
4	Training loss [a]	0.329	0.187	0.062	0.120
	Validation loss [b]	0.335	0.197	0.083	0.144
	SMILES reconstruction [c]	0.0%	1.4%	13.6%	0.1%
	Character reconstruction [d]	87.6%	92.5%	96.5%	92.9%
8	Training loss [a]	0.301	0.131	0.071	0.232
	Validation loss [b]	0.306	0.140	0.090	0.252
	SMILES reconstruction [c]	0.4%	6.6%	12.1%	0.0%
	Character reconstruction [d]	88.7%	94.7%	96.3%	88.0%
16	Training loss [a]	0.284	0.105	0.018	0.005
	Validation loss [b]	0.290	0.113	0.035	0.022
	SMILES reconstruction [c]	1.2%	13.5%	61.9%	76.5%
	Character reconstruction [d]	89.4%	95.8%	98.9%	99.4%
32	Training loss [a]	0.264	0.101	0.020	0.232
	Validation loss [b]	0.265	0.105	0.037	0.253
	SMILES reconstruction [c]	1.4%	17.0%	57.4%	0.0%
	Character reconstruction [d]	90.4%	96.1%	98.7%	87.9%
64	Training loss [a]	0.253	0.118	0.004	**0.000**
	Validation loss [b]	0.258	0.125	0.015	**0.006**
	SMILES reconstruction [c]	1.3%	11.2%	84.9%	**94.9%**
	Character reconstruction [d]	90.6%	95.3%	99.6%	**99.9%**

[a] The loss evaluated against the training data set after 100 epochs.
[b] The loss evaluated against the validation data set after 100 epochs.
[c] The percentage of SMILES in the holdout data set which were completely reconstructed.
[d] The percentage of characters in SMILES in the holdout data set which were successfully reconstructed.

Bibliography

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[2] James O. Berger. *Statistical Decision Theory and Bayesian Analysis*. Springer-Verlag, New York, 2nd edition, 1985.

[3] Regine S. Bohacek, Colin McMartin, and Wayne C. Guida. The art and practice of structure-based drug design: A molecular modeling perspective. *Medicinal Research Reviews*, 16(1):3–50, 1996.

[4] François Chollet et al. Keras. https://keras.io, 2015.

[5] Lynda B. M. Ellis, Dave Roe, and Lawrence P. Wackett. The University of Minnesota Biocatalysis/Biodegradation Database: the first decade. *Nucleic Acids Research*, 34(suppl_1):D517–D521, 01 2006.

[6] Kathrin Fenner, Junfeng Gao, Stefan Kramer, Lynda Ellis, and Larry Wackett. Data-driven extraction of relative reasoning rules to limit combinatorial explosion in biodegradation pathway prediction. *Bioinformatics*, 24(18):2079–2085, 07 2008.

[7] Karl Pearson F.R.S. Liii. on lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 2(11):559–572, 1901.

[8] Sabrina Jaeger, Simone Fulle, and Samo Turk. Mol2vec: Unsupervised machine learning approach with chemical intuition. *Journal of Chemical Information and Modeling*, 58(1):27–35, 2018.

[9] Artur Kadurin, Alexander Aliper, Andrey Kazennov, Polina Mamoshina, Quentin Vanhaelen, Kuzma Khrabrov, and Alex Zhavoronkov. The cornucopia of meaningful leads: Applying deep adversarial autoencoders for new molecule development in oncology. *Oncotarget*, 8(7), December 2016.

[10] Sunghwan Kim, Jie Chen, Tiejun Cheng, Asta Gindulyte, Jia He, Siqian He, Qingliang Li, Benjamin A Shoemaker, Paul A Thiessen, Bo Yu, Leonid Zaslavsky, Jian Zhang, and Evan E Bolton. PubChem 2019 update: improved access to chemical data. *Nucleic Acids Research*, 47(D1):D1102–D1109, 10 2018.

[11] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[12] Diederik P Kingma and Max Welling. Auto-Encoding Variational Bayes. *arXiv e-prints*, page arXiv:1312.6114, Dec 2013.

[13] Günter Klambauer, Thomas Unterthiner, Andreas Mayr, and Sepp Hochreiter. Self-normalizing neural networks. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 971–980. Curran Associates, Inc., 2017.

[14] Greg Landrum et al. Rdkit: Open-source cheminformatics, 2006.

[15] E.L. Lehmann and G. Casella. *Theory of Point Estimation*. Springer Texts in Statistics. Springer New York, 2006.

[16] Leland McInnes, John Healy, Nathaniel Saul, and Lukas Grossberger. Umap: Uniform manifold approximation and projection. *The Journal of Open Source Software*, 3(29):861, 2018.

[17] H. L. Morgan. The generation of a unique machine description for chemical structures-a technique developed at chemical abstracts service. *Journal of Chemical Documentation*, 5(2):107–113, 1965.

[18] Hideaki Nojiri, Hiroshi Habe, and Toshio Omori. Bacterial degradation of aromatic compounds via angular dioxygenation. *The Journal of General and Applied Microbiology*, 47(6):279–305, 2001.

[19] PhRMA. 2014 biopharmaceutical research industry profile. Technical report, Pharmaceutical Research and Manufacturers of America, 2014.

[20] Mariya Popova, Olexandr Isayev, and Alexander Tropsha. Deep reinforcement learning for de-novo drug design. *Science Advances*, 4, 11 2017.

[21] Danilo Jimenez Rezende and Shakir Mohamed. Variational inference with normalizing flows, 2015.

[22] David Rogers and Mathew Hahn. Extended-connectivity fingerprints. *Journal of Chemical Information and Modeling*, 50(5):742–754, 2010. PMID: 20426451.

[23] Lars Ruddigkeit, Ruud van Deursen, Lorenz C. Blum, and Jean-Louis Reymond. Enumeration of 166 billion organic small molecules in the chemical universe database gdb-17. *Journal of Chemical Information and Modeling*, 52(11):2864–2875, 2012. PMID: 23088335.

[24] Marwin H. S. Segler, Thierry Kogej, Christian Tyrchan, and Mark P. Waller. Generating Focussed Molecule Libraries for Drug Discovery with Recurrent Neural Networks. *arXiv e-prints*, page arXiv:1701.01329, Jan 2017.

[25] Teague Sterling and John J. Irwin. Zinc 15 – ligand discovery for everyone. *Journal of Chemical Information and Modeling*, 55(11):2324–2337, 2015. PMID: 26479676.

[26] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 3104–3112. Curran Associates, Inc., 2014.

[27] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9:2579–2605, 2008.

[28] David Weininger. Smiles, a chemical language and information system. 1. introduction to methodology and encoding rules. *Journal of Chemical Information and Computer Sciences*, 28(1):31–36, 1988.

[29] Jörg Wicker, Kathrin Fenner, Lynda Ellis, Larry Wackett, and Stefan Kramer. Predicting biodegradation products and pathways: a hybrid knowledge- and machine learning-based approach. *Bioinformatics*, 26(6):814–821, 01 2010.

[30] Jörg Wicker, Tim Lorsbach, M. Gütlein, Emanuel Schmid, Diogo Latino, Stefan Kramer, and Kathrin Fenner. Envipath - the environmental contaminant biotransformation pathway resource. *Nucleic Acids Research*, 44, 11 2015.

[31] R. J. Williams and D. Zipser. A learning algorithm for continually running fully recurrent neural networks. *Neural Computation*, 1(2):270–280, June 1989.