School of Computing

FACULTY OF ENGINEERING
AND PHYSICAL SCIENCES

UNIVERSITY OF LEEDS

Final Report

Antibiotic Discovery Using Generative AI

Jamal Ahmed

Ryan Chung

Nicholas Matthew Dhamardi

Mark Stephen Mulila Muinde

Alper Onder

Mitali Sen

Submitted in accordance with the requirements for the degrees of

MEng, BSc Computer Science

MEng, BSc Computer Science with Artificial Intelligence

2022/23

Jamal Ahmed

Ryan Chung

Nicholas Matthew Dhamardi

Mark Stephen Mulila Muinde

Alper Onder

Mitali Sen

(Signatures of students)

TABLE OF CONTENTS

Acknowledgements

We would like to thank our supervisor, Dr. Nishant Ravikumar, for his support and guidance throughout the duration of this project. His advice and suggestions were crucial to our process and allowed us to develop a novel and interesting piece of research.

Additionally, we would like to thank the project coordinator, Dr. He Wang, for his advice and enthusiasm throughout the project whilst also managing several other groups. His wide area expertise was especially beneficial for this project as it pivoted us in the right direction, making our ambitious goals more tractable.

Contributions

Names	**Jamal Ahmed**	**Ryan Chung**	**Nicholas Dhamardi**	**Alper Onder**	**Mark Muinde**	**Mitali Sen**
Contributions	- Data quality and test sets - Random Forest classifier - Sequential classifier - Classifier metrics - Contextual DRNN - Conditional DRNN - Generative Examiner Network - Generator metrics - SMILES to PDB - Report Writing and GEN diagram - Research into and giving ideas for solutions into the problem space - Gitlab pruning	- Chairing Meetings - Report Writing - Molecular Docking - External Screening - Examiner - PDBQT Conversions and Pre-Processing - Naive Bayes - Zero R - SGD for Text - GNN Classifier - Research into and giving ideas for solutions into the problem space	- Report Writing - XGBoost -Variable Auto Encoder (VAE) Feasibility Testing - Google drive/ Gitlab repository management - Report Quality Assessment - Research into and giving ideas for solutions into the problem space	- RNN Classifier -MLP GAN Feasibility Testing - SMILES GAN - SELFIES GAN - Report writing - Research into and giving ideas for solutions into the problem space	- Taking Minutes - Dataset Search & Acquisition - Dataset Preprocessing - Dataset Quality Assessment - Dataset Building & Evaluation - Report Writing and formatting - Managing Google Drive - Research into and giving ideas for solutions into the problem space	- Taking Minutes - Report Writing and Editing - Data Acquisition and Evaluation - MLP Classifier - Chemical Data analysis - Report Planning, Organisation and Delegation - Research into and giving ideas for solutions into the problem space -Collecting dataset of approved drugs

Summary

This project investigates machine learning applications in classifying and generating molecular structures with drug-like properties, specifically targeting antibacterial activity against Escherichia coli (E. coli). It employs SMILES and SELFIES representations, molecular descriptors, and fingerprints to investigate the feasibility of discovering molecules with desired characteristics.

A balanced dataset of approximately 120,000 molecules was compiled from ChEMBL, ECMDB, DrugBank, and Repurposing Hub. Various input types were used for different classifier models, with random Forest (RF) emerging as the top-performing model, achieving 89% accuracy on 20% stratified and 20% random test sets, as well as in 10-fold cross-validation.

Several generative models were developed, comprising Deep Recurrent Neural Networks (DRNN), Generative Adversarial Networks (GAN), and Generative Examiner Networks (GEN). The GANs were trained with both SELFIES and SMILES representations, with the SELFIES GAN emerging as the superior model due to the robustness of the SELFIES representation. Despite this, DRNNs outperformed the GANs in terms of validity, uniqueness, and execution speed.

Three DRNNs and a subsequent GEN model were developed. These include the E. coli RNN, Contextual RNN, Conditional RNN, and, when paired with an examiner network, a Contextual GEN. The E. coli RNN was trained solely on E. coli-active molecules, while the Contextual RNN introduced an [ECOLI] token for E. coli-active sequences. The Conditional RNN utilised time-distributed secondary inputs for the molecular class/RF-derived scores. The Examiner Networks integrated the RF with other Quantitative Structure-Activity Relationship (QSAR) models, like toxicity prediction and synthetic accessibility, to retrain the generator using generated molecules.

Most models, pre and post examination, generated theoretically valid, novel, and unique sequences, attributable to the more robust SELFIES format, which is readily convertible to SMILES and molecular objects. These molecules underwent external QSAR analysis, such as molecular docking, to evaluate binding affinity to E. coli proteins which can be indicative of antibacterial activity. Nevertheless, further experimental testing is needed to confirm their predicted antibacterial properties, safety, and efficacy, ensuring practical applicability in drug development.

This project offers societal benefits by potentially enabling the computational synthesis of novel molecules targeting specific activities, leading to the discovery of new antibacterial agents against E. coli. This can be extended to other pathogens. These findings could potentially inspire the development of cutting-edge computational tools in the field, making drug discovery more efficient and accessible.

1. Introduction

1.1 Introduction

One of the biggest threats in global health, food security and development is antibiotic resistance (WHO, 2020), making us not only more vulnerable to bacterial disease such as pneumonia or tuberculosis, but also to a number of viral ones, such as influenza and Covid-19. In an ongoing battle against one of the world's most widespread pandemics, and with a rising global population, the need for the accelerated development and delivery of life-saving drugs is greater now more than ever.

The growing resistance of bacteria against antibiotics has rendered many of these drugs ineffective, with analogous methods for drug discovery struggling to keep pace with this accelerated growth (Coates et al., 2011; Livermore, 2004). As a result of this, the last three decades have been termed as an "innovative void" (*ReAct*, 2021) when it comes to the discovery of antibiotics. With easier-to-synthesise antibiotics having been discovered early on, the latest discoveries are all borne of the dereplication problem: the repeated discovery of existing antibiotic classes and their variants, which lack the effectiveness to properly combat increasing antibiotic resistance (Cox et al., 2017, cited in Stokes et al., 2020). What is needed is the discovery of a novel class of antibiotics – a complex and challenging task (Coates et al., 2011).

To discover a new class of drugs that is structurally and chemically unique to the existing classes, we need to expand our search beyond all currently known compounds. With an estimated 10^{60} compounds within it, exploring the entirety of the chemical space is a demanding task (Bilodeau et al., 2022). This is where generative models come in. Generative models allow us to navigate the entire chemical space at an accelerated pace by leveraging deep learning to tackle the inverse molecular design problem:

"Given a set of desired properties, what is the set of molecules that will satisfy those properties?" (Bilodeau et al., 2022)

So what are these desired properties? In the case of antibiotics, it is desired that the drugs exhibit high solubility, low toxicity, and broad-spectrum potency (Bilodeau et al., 2022; Kellis & Jin, 2021). As one can imagine, finding such drugs is no simple feat, and to do so via traditional methods of relying purely on human expertise is time-consuming and costly. Computer scientists are especially disadvantaged at understanding the biological aspects to this problem and at recognising all the possible patterns found in the structures of chemical compounds. Generative models, however, present "the key to success: automatic feature learning" (Kellis & Jin, 2021). Given the right model, training data, and optimisation parameters, a classifier could be trained to accurately determine whether a given compound displays the desired antibiotic properties. A generative model could subsequently leverage guidance from this classifier and use automatic feature learning to predict molecules with antibacterial activity. It was in a similar manner, using a deep neural network to target the gram-negative bacteria *E. coli*, that Stokes et al (2020) discovered the novel and structurally divergent *Halicin* — an antibiotic which showed strong bactericidal activity against a variety of pathogenic bacteria, including the pan-resistant *A. baumannii*.

This project aims to train such a classifier and produce a generative model based on deep learning techniques with the ambitious goal of discovering more effective antibiotics. A further aim is to verify the authenticity of these results using QSAR-informed evaluation techniques such as toxicity screening, synthetic accessibility screening, and molecular docking. Results will be presented, and the success of the project discussed, as well as a critical assessment of its shortcomings and the scope they provide for future work.

1.2 Project Aim

The aim of this project is to produce a machine learning model that can generate novel molecules that display antibiotic properties that target the bacteria *Escherichia coli* (E. coli). This will answer the question: Is it feasible to generate new chemical compounds with antibiotic properties using machine learning methods?

1.3 Project Objectives

The objectives of the project are as follows:

a) Conduct adequate, comprehensive background research into computer-aided antibiotic design and generation so as to establish benchmarks from which we can develop our project. This includes which databases to use, which input formats perform best in this scenario and which classification models have shown promise in the problem domain.
b) Curate and create a set of both drugs with antibiotic properties that target E. coli as well as an equal number of those that do not so as to keep the dataset balanced.
c) Develop and experiment with different classification models so as to determine the best performing classifier for E. coli antibiotics.
d) Develop a generative model that uses the correctly classified E. coli drugs to create new variations that may serve as new antibiotics for E. coli.
e) To compile an academic report that comprehensively describes the final model and the steps taken to produce it.

1.4 Project Deliverables

DELIVERABLE	**LOCATION**
Link to project repository containing all files, code scripts and any material used to develop the classifier and generator. The repository will also contain formal documents such as minutes from group meetings.	https://gitlab.com/MarkMuinde/comp5530m
Jointly authored group report regarding the process of attempting the project, including background research, design and framework, implementation, validation/evaluation, conclusions and future work.	This report
Individually authored reports regarding technical and non-technical learning in the project, challenges, successes and failures.	Submitted individually via the student portal
PDBQT files of potential molecules for further analysis to determine efficacy of functioning as an antibiotic for E. coli	https://gitlab.com/MarkMuinde/comp5530m

1.5 Legal, Social, Ethical, and Professional Issues

It is acknowledged that there is the possibility that the results of this project may be used in drug discovery. As such, it is clearly stated that any compound combinations generated by this project are not approved for human use as they have not been scrutinised by any medical standards bodies.

As this project's research involves discovering new drugs, there was the possibility of encountering compounds which are patented/copyrighted. As such, great care was taken to ensure that the data used in training the classification model was open-source and publicly available. Where this was not the case, the owner of a dataset was contacted and express permission to use their dataset for academic purposes was acquired.

The use of machine learning may raise ethical and social concerns about the accountability of the software developers involved in *in-silico* drug discovery. For example, if parties make use of drugs discovered by AI models, who bears the burden of responsibility should any injuries be incurred? What safeguarding precautions should developers take to ensure drugs are properly tested before being released to the public? There is also the potential for these AI models to be used for objectives other than medical purposes. For example, to discover chemicals with similar effects to illicit substances, in order to circumvent the law. Should legal measures be put in place to prevent these models from being exploited in this manner?

Additionally, public perception of AI-driven drug discovery could be an issue. As such, ensuring the reliability and safety of these drugs is of utmost concern. Any drugs generated by AI models should undergo rigorous testing by professionals with domain expertise and should not be used without certification from a licensed medical body.

2. Background Research

2.1 Molecular Representations

The quality and quantity of data can make or break the effectiveness of any machine learning algorithm. But how can information be translated, such that a computer can use it? For numerical problems, one can simply enter the numbers into the computer. For image problems, each individual pixel is represented by numerical values. But for a problem involving molecules, it is not as trivial to represent chemical information in a concise, useful format for a computer to use.

2.1.1 Graphs

Some formats used to electronically represent molecules are "built on the molecular graph representation". (David et al., 2020, p. 2) In their paper, "Molecular representations in AI-Driven Drug Discovery: A review and practical guide", David et al. state that "the idea behind the molecular graph representation lies in mapping the atoms and bonds that make up a molecule into sets of nodes and edges" (2020, p. 2) .The atoms would be represented by the nodes in the graph, and the bonds would be represented by the edges. However, there is a difference between a regular graph and a chemical graph. Normally, circles are used to represent a node in a graph. In a chemical graph, nodes are either represented by their atomic letter from the periodic table, or in the case of carbon atoms, nodes are represented by a connection of two edges. (David et al., 2020).

Nonetheless, a computer cannot use a molecular graph in its raw form to perform computation. In order to do this, the nodes and edges must be converted into a format that the computer can interpret. One way of doing this is to use matrices. For example, a chemical graph could be represented by three matrices: an adjacency matrix, a node feature matrix, and an edge feature matrix. The adjacency matrix contains information on which nodes are connected. The node feature matrix contains information on the atom the node represents, and the edge feature matrix contains information on the bonds that are connecting the nodes. In this format, a computer can then traverse these graphs using various graph traversal algorithms, such as breadth-first search and depth-first search. (David et al., 2020)

2.1.2 SMILES

Another format that can be used to represent chemical graphs for use in computation is the Simplified Molecular-Input Line-Entry System (SMILES). According to Krenn et al., SMILES has "become the *de facto* standard representation in cheminformatic" (2022, p. 4). In this format, the chemical compounds are defined as a string. The atoms in the compounds are represented by letters. Branches and rings in the molecule can also be represented in this format, as well as properties of the compound. (Krenn et al., 2022). Each atom in the compound is also given a number. The number determines the order of traversal when using a graph traversing algorithm on the format (David et al., 2020). While SMILES strings are a convenient format to use due to their popularity and minimal storage requirements (David et al., 2020), they are not always unique to a compound and may produce invalid graphs (Krenn et al., 2022).

2.1.3 SELFIES

One possible solution to the drawbacks presented by SMILES is SELf-reFerencIng Embedded Strings, or SELFIES – a format that only allows for the creation of strings that have a corresponding chemical graph. This means that only chemically viable molecules can be produced using this representation. A molecule can have multiple SELFIES representations. SELFIES is a formal grammar. The derivation rules of the grammar were designed such that no molecules with syntax or semantic errors can be produced, ensuring that every SELFIES string is a viable molecule (Krenn et al., 2022). SELFIES offer an attractive solution as they are more robust than SMILES. However, they cannot yet represent certain molecules like macromolecules or molecules with complicated bonds (Krenn et al., 2022).

2.1.4 Molecular Fingerprints

A key molecular representation to consider is molecular fingerprinting, i.e., encoding the structural characteristics of a molecule into a bit vector. Molecular fingerprints are essential in the field of cheminformatics by allowing for quick comparisons in structural similarities among different molecules and have thus proven useful in QSAR studies (Capecchi et al., 2020). An example of a molecular fingerprint is the RDKFingerprint. It is a rule-based fingerprint, which means that the paths or topology between atoms and bonds within molecular substructures is taken into account (Zagidullin et al. 2021).

2.2 Physicochemical Properties, Processes, and Interactions

2.2.1 QSAR Modelling and Molecular Descriptors

Molecular descriptors are a quantitative way of representing the various physical and chemical properties such as the size, solubility, and polarity of a compound (Guha & Willighagen, 2012). As such, they have proven to be a useful tool in the field of cheminformatics, and more importantly, drug discovery.

QSAR modelling is a predictive technique used for correlating the biological activity of a compound with the quantitative representation of its structural features, i.e. its molecular descriptors (Li. et al. 2021). The technique has been used successfully in many applications including that of antibiotic drug discovery. Accordingly, this technique is believed to be fitting for the project use case and has thus been included by adding a number of molecular descriptors as features to the classifier model. Examples of the descriptors one can include to improve the amount of stereochemical information available to model are the molecular weight, the number of hydrogen bond donors and acceptors, and the number of rotatable bonds of a molecule. Additionally, the polarity of a molecule can be considered using its lipophilicity (logP), i.e. the preference of a compound to dissolve in an organic solvent (*REVIVE*, n.d.), and the topological polar surface area (TPSA) of a molecule – a convenient measure of the surface area of a molecule that avoids performing 3D calculations (Prasanna et al. 2009).

Finally, one should consider the Quantitative Estimate of Drug-likeness, or QED – the empirical rationale of which reflects the underlying distribution of the above-mentioned molecular descriptors and more (The RDKit Documentation, 2022). In the absence of any structural resemblance to an approved drug, QED provides us with a useful guideline in early-stage drug discovery by estimating how well a compound would perform as a drug based on factors such as its permeability or solubility (Bickerton et al., 2012).

2.2.2 Mechanisms of Actions

In chemistry, a mechanism (or mode) of action can be defined as the way in which a drug or compound produces an effect in the body (National Cancer Institute, n.d.). For an antibiotic, a mode of action (MOA) can be considered as its antimicrobial potency and involves targeting "a unique feature of the bacterial structure or their metabolic processes" (Etebu & Arikekpar, 2016, p. 96). Some of the most common MOA for antibiotics are as follows:

a) **Inhibition of protein synthesis:** Proteins are responsible for a host of metabolic and physiological processes, i.e. processes that are essential life processes in all living organisms. Given their importance, any disruption of protein synthesis in a bacterial cell would inhibit its growth, or even kill it completely. Antibiotics that inhibit protein synthesis do so by targeting the factory for protein synthesis in the cell: ribosomes.
b) **Inhibition of nucleic acid synthesis:** The pathways that result in the synthesis of nucleic acids are integral to the survival and posterity of bacterial cells. Inhibiting these is done by blocking the synthesis and replication of DNA in bacteria. Antibiotics which target the synthesis of nucleic acids, such as quinolones, target the DNA gyrase enzyme in bacterial cells. This adversely affects bacterial RNA polymerase but does not interact with mammalian RNA polymerase, making these drugs effective antibacterials (Etebu & Arikekpar, 2016).

The uniqueness of these processes provides practical information that can be used in identifying antibiotic properties in a compound. Specifically, they can be used to determine the binding affinities between antibiotics and bacterial cell proteins through molecular docking, an evaluation technique which is discussed in Section 2.6.1. For example, ciprofloxacin is a quinolone that targets the DNA-gyrase protein in a bacterial cell and thus inhibits nucleic acid (DNA) synthesis. Similarly, rifampicin is a broad-spectrum antibiotic that targets the DNA-directed RNA-polymerase protein in a bacterial cell, which is a binding site on the cell's ribosomes (Etebu & Arikekpar, 2016; Hamouche et al., 2021). As such, the binding of rifampicin E. coli RNA polymerase enzymes leads to the inhibition of protein synthesis in E. coli, making rifampicin an effective antibiotic.

2.3 Cheminformatics Toolkits

2.3.1 Open Babel

OpenBabel is an open-source cheminformatics 'toolkit' that facilitates searching, conversion, analyses and storage of over 100 representations of cheminformatics. These formats include SMILES and molFiles, among others (O'Boyle et al., 2011). Through such a big store of information, translation of data from one format to another is simple and direct, therefore facilitating deeper insights and understanding into cheminformatics research. It can also be used to convert PDB files to PDBQTs, i.e the required format of files used for molecular docking. This allows the addition of explicit hydrogens and partial charges.

2.3.2 RDKit

RDKit is considered to be the most well rounded comprehensive toolkit available for cheminformatics as it provides features that are both present in OpenBabel and DeepChem whilst also integrating with standard libraries such as numpy and matplotlib for visualisation (The RDKit Documentation, 2022). Various molecular fingerprints can be generated from the chemical structure input which enables efficient molecular similarity estimations and substructure searching of chemicals. Topological, geometrical and physicochemical descriptors are all available through this library and are useful for tasks such as virtual screening, QSAR modelling, and molecular property prediction. There is also support for 2D and 3D molecular coordinate generation and conformational sampling which is helpful for tasks like molecular docking and structure based drug design.

2.4 Classification Models

Classification models have been a powerful tool in the domain of in-silico drug discovery based on several sources (Badura et al., 2020; Feng et al., 2019; Zoffman et al., 2019), including the study where the novel antibiotic *Halicin* was discovered by Stokes et al. (2020) using a binary classifier. These papers discussed using a variety of features discussed in previous subsections such as molecular fingerprints and the MOA, as well as models like a Random Forest (Ahn et al., 2022; Zoffman et al., 2019) for classification. It is based on these studies that the following classification models have been chosen for experimentation and to determine the optimal one for the project use case.

Some of the most seminal machine learning models today such as a Random Forest (Brownlee, 2021), a Multilayer Perceptron (Bento, 2021), and a Naive Bayes classifier (Frank et al., 2009) were implemented. Alongside these, an SGD classifier – which uses stochastic gradient descent for linear classification – and a ZeroR classifier – a simple text classifier which predicts the most common class label (e.g. active or inactive towards E. coli) for a given SMILES string (Frank et al., 2009) – were included. Due to the RF's algorithm's propensity for classification of tabular data (Borisov et al., 2022), such as molecular fingerprints, an XGBoost classifier – a tree-based ensemble method which has been improved by a gradient boosting algorithm (Mello, 2020) – has also been included. Finally, two neural network models were implemented. These were a Recurrent Neural Network to process sequential data (Brownlee, 2019), such as SMILES strings, and a Graph Neural Network (GNN) to use molecular graphs directly as the input, similar to the one created by Stokes et al. (2020).

2.5 Generative Models

Generative models are powerful tools for generating new data samples based on learned patterns from existing data (Goodfellow et al., 2014; Kingma & Welling, 2013). These models have gained traction in drug discovery as they offer the potential to efficiently generate novel molecules with desirable properties and optimise existing molecules for improved efficacy or safety (Gómez-Bombarelli et al., 2018). By exploring vast and complex chemical spaces, generative models can accelerate the discovery process and reduce the time and resources required compared to traditional methods (Stokes et al., 2020).

2.5.1 Generative Adversarial Networks

Generative Adversarial Networks (GANs) are a class of generative models in machine learning, introduced by Goodfellow et al. (2014). GANs consist of two neural networks, a generator and a discriminator, that engage in an adversarial training process. The generator creates synthetic data samples, while the discriminator evaluates their authenticity. Over time, the generator improves its ability to generate realistic samples, ultimately producing data that is near-indistinguishable from real examples. In drug discovery, GANs have shown potential for generating novel molecules with desired properties and optimising existing molecules for better efficacy or safety (Gupta et al., 2018; Olivecrona et al., 2017). By exploring vast chemical spaces and generating chemically valid and synthesizable molecules, GANs offer a promising approach to accelerate the drug discovery process and reduce reliance on traditional, time-consuming methods (Kadurin et al., 2017).

2.5.2 Recurrent Neural Networks

Recurrent Neural Networks (RNNs) are a class of artificial neural networks designed to handle sequential data by maintaining a hidden state that can capture information from previous time steps (Hochreiter & Schmidhuber, 1997). Deep Recurrent Neural Networks (DRNNs) are an extension of RNNs with multiple layers of hidden units, enabling the learning of more complex and abstract representations of the input data (Pascanu et al., 2013). When working with SMILES strings, RNNs and DRNNs can be utilised to model the sequential nature of the textual representations of chemical structures. By learning from the SMILES strings, these networks can predict various molecular properties, such as biological activities or physicochemical properties (Goh et al., 2017). In the context of antibiotic discovery or classification, RNNs and DRNNs can be employed to process SMILES strings and predict antibacterial activities of molecules, enabling the rapid identification of potential antibiotic candidates and accelerating the discovery process (Stokes et al., 2020).

2.5.3 Generative Examiner Networks

Generative Examination Networks (GENs) are a type of deep generative model in which the generator is periodically examined by an independent examiner network during training (van Deursen et al., 2020). The generator produces sequences that mimic the underlying distribution of the training data while the examiner acts as a screening model that evaluates the quality of the generated data. Unlike a GAN where both the discriminator and the generator provide each other with feedback as peers would, GENs utilise a student-teacher dynamic where the feedback process is unidirectional; only the generator adjusts its parameters to improve the quality of the generated data based on the examiner's feedback. This allows the generators to incorporate prior knowledge with the examiner's feedback into the generation process, making them a flexible and powerful tool for generative problems. As such, they have led to previously successful generative models for in-silico drug design.

Van Deursen et al. (2020) presented this novel approach to train deep generative networks for SMILES generation by using a GEN composed of an online DRNN generator and an independent online examination network. The results showed the best-performing architecture used one bi-Long-Short-Term-Memory(biLSTM) embedding layer followed by 4 parallel concatenated biLSTM encoding layers. Overally, the models achieved high validity rates (95-98%) and novelty rates (85–90%) whilst generating SMILES with strong conservation of the property space (95–99%).

Another innovative approach for improved *de novo* drug design was presented by Perron et al. (2022) using a DRNN and an examiner network, as well as domain-specific knowledge involving QSAR modelling techniques. Additionally, an optimisation strategy called "Hillclimb-MLE" (Perron et al., 2022, p. 694) was employed to iteratively update a set of SMILES based on a scalar reward function that combined 13 targets, including probabilities of activity, similarity to the project dataset, and the QED. This approach resulted in the generation of valid, diverse, and drug-like molecules with optimised properties. Notably, the generative model was able to propose innovative solutions within the chemical space, even with limited data to learn from. This work showcases the potential of integrating deep learning techniques, such as GENs, and domain expertise to tackle complex problems in drug discovery and optimisation.

2.5.5 Generator Metrics

To properly assess the accuracy and effectiveness of the developed generative models, robust evaluation metrics need to be included. As such, the following metrics have been included as they are considered to be critical in the assessment of the generative models designed.

2.5.5.1 Validity

In van Deursen et al. (2020), a generated molecule was considered to be valid if it could be successfully converted into an InChIKey. In this project, the validity of a molecule refers to its ability to be converted into a molecular fingerprint. To calculate the percentage of valid molecules generated by a model, the number of valid molecules is divided by the total number of generated SMILES or SELFIES.

2.5.5.2 Uniqueness

The uniqueness of a molecule is determined by whether or not duplicates of it are produced during generation. As such, the percentage of unique molecules produced is considered to be the ratio of the number of unique molecules, i.e. non-duplicated molecules, to the number of valid molecules generated (Deursen et al., 2020).

2.5.5.3 Novelty

The novelty of a molecule can be defined by whether or not it is present in the training set (Grisoni et al., 2020, cited in Grant and Sit, 2021). If a generated molecule is not present in the training set, it is considered to be novel, however, another type of novelty can be considered as well – external novelty. A molecule is considered externally novel if it cannot be found in external datasets, i.e. datasets that are distinct from the training set. For example, the PubChem database is used as an external dataset.

2.5.5.4 Binding Affinity

Binding affinity can be defined as “the strength of the binding interaction between a single biomolecule (e.g. protein or DNA) to its ligand/binding partner (e.g. drug or inhibitor)” (Malvern Panalytical, n.d.). In this project, the generated molecules act as the ligands, and the chosen E. coli proteins act as the receptors to these ligands.

2.5.5.5 Toxicity and Synthetic Accessibility Screening

Two key factors to consider for in-silico screening are toxicity testing and synthetic accessibility, i.e. the “process to assess the ease of synthesis of compounds” (Baba et al., 2018, p. 217). To do so, the model presented by Pu et al. (2019) in their paper titled “eToxPred: a machine learning-based approach to estimate the toxicity of drug candidates” has been used. The *e*ToxPred model uses a Deep Belief Network (a generative probabilistic model) to predict the synthetic accessibility score of compounds, and an Extremely Randomized Trees algorithm, which is “a new tree-based ensemble method for supervised classification and regression problems” (Geurts et al., 2006, p. 3), to predict the toxicity of potential drugs.

2.6 In Silico Screening

In Silico screening refers to the use of Computer Aided Drug Design (CADD) methods to develop, test and validate pharmacological hypotheses. This includes the use of data mining, data analyses, data science, machine learning, and molecular docking among other computer-based tools (Ekins et al., 2007). Some of the tools are discussed below.

2.6.1 Molecular Docking

Molecular docking refers to simulating molecular interactions so as to predict binding modes and affinity between receptors and ligands (Fan et al., 2019). This method can be used to run multiple concurrent simulations in order to determine the feasibility of the screened drugs. There are three main types of molecular docking:

a) Rigid docking: The position of the molecule changes but the shape remains rigid.
b) Flexi-rigid/Flexible-rigid: Small ligand molecules conform to the receptor's shape.
c) Flexible/soft docking: Ligands and receptors are flexible. The most accurate type, but has heavy computational overheads.

An example of a molecular docking software is Autodock Vina. It is an open-source software program used to perform molecular docking as discussed above. It's available as part of the AutoDock Suite docking engines and offers higher prediction accuracies and can take advantage of multithreading, therefore offering faster performance (Trott & Olson, 2009).

2.7 Libraries and Toolkits

Both the classification models as well as the generative models were designed using a variety of Python libraries and toolkits such as Pandas, Numpy, PyTorch, and Scikit-Learn. Additionally, the open-source data mining software known as WEKA was used. This was primarily used for the text classification models as WEKA's in-built text preprocessing allowed for simple but effective implementations of SMILES strings classifiers.

3. Project Framework and Data Characterisation

3.1 Project Framework

This project can be split into two main elements: the classification and the generation problems. The approach used to solve these two problems is described below.

Firstly, the feasibility of different machine learning algorithms was tested for use in the examiner network. In this phase, three different types of chemical representations were trialled: SMILES, graphs, and molecular descriptors. Several different types of classification algorithms were also trialled – these will be described later in the report. Results were evaluated using cross-fold validation and two tests.

For the generation problem, two main types of networks were trialled: GANs and GENs. The GEN used the best performing classifier from the first phase as the examiner. In addition to this, two types of chemical representations were also tried: SMILES and SELFIES. The networks were evaluated on several metrics, such as novelty, toxicity, and validity of generated tokens. Molecular docking was subsequently performed on the generated molecules in order to determine their binding affinities to key proteins in E. coli. The final model was the generative model that achieved the best scores on the chosen metrics.

3.2 Data Sources: Collection and Cleaning

The data sources used are open-source medical databases that contain information on drugs which target E. coli. Where a licence was required, the necessary permissions were acquired from the copyright owners as seen in Appendix A. These databases store drugs in the form of SMILES strings.

3.2.1 Database Selection

Following Melo et al., 2021, a number of general drug discovery and bioinformatics databases were discovered. Upon further investigation, it was discovered that one of the databases, namely ZINC15, contained almost all the data from the other databases. An attempt was made to download the entire database at once but it was too large. Therefore, it was decided to peruse the ZINC15 catalogue so as to cherry-pick the best data sources for the project use-case.

The datasets chosen were ChEMBL, DrugBank, Drug Repurposing Hub, and E. coli Metabolome Database (ECMDB), resulting in a large dataset of 124,055 SMILES.

3.2.2 Dataset Preparation

The datasets used in training the classifier was prepared by the following steps:

a) Downloading a dataset from a selected database.
b) Where necessary, converting the downloaded data from its source file format to a comma-separated-value (.csv) or excel workbook (.xlsx) for data manipulation.
c) Where necessary, isolating the SMILES from the downloaded dataset that regard E .coli antibiotics.
d) Getting an equal sample of non-E. coli antibiotic SMILES so as to have a balanced dataset.
e) Exporting the data in (c) and (d) as .xlsx workbooks.

f) Once steps (a) to (e) were done for all the selected databases, all E. coli SMILES were combined into one .xlsx workbook, all non-E. coli SMILES were combined into another .xlsx workbook, and one more .xlsx workbook was made which contained all E. coli and non-E. coli SMILES.
g) The combined workbooks in (f) were given a new one-hot encoded column that would indicate whether the SMILES refers to an E. coli antibiotic drug or not.

Appendix A illustrates the step-by-step preparation process for each dataset. Step (b) was performed using custom scripts as described in Appendix A. Steps (c) to (g) were performed using Alteryx Designer as described in Appendix A. Additionally, a dataset containing only approved drugs was collected for the comparison of their molecular structures with those of the molecules generated for this project.

3.2.3 Dataset Characterisation

The datasets resulting from the preparation process in Section 3.2.2 had the following fields:

a) ID: the unique identification number of a drug as obtained from the relevant database. It is a String field, nominal by data type and semantics.
b) SMILES: the SMILES of a particular compound. This column is also unique. It is a String field, nominal by data type and semantics.
c) ECOLI STATUS: A one-hot encoded column indicating whether the SMILES refers to an E. coli antibiotic. ‘1’ means yes, ‘0’ means no. It is a numerical field by data type but categorical by semantics.

3.3 Data Variables and Features

3.3.1 SMILES & SELFIES

While SMILES strings are considered to be the *de facto* standard in computational chemistry, SELFIES strings are a more robust representation as each SELFIES string corresponds to a valid molecule and can be used in any machine learning model (Krenn et al., 2020). While SMILES-based generators can often produce invalid molecules, SELFIES generators generally do not. Since both forms are string representations of molecules, atoms are represented by the same symbols (letters) as in the periodic table. However, they are syntactically different from each other as well as a typical chemical formula. Figure 3.1 depicts the rules for deriving the syntax for both these representations.

3.3.1.1 String Tokenisation for Generation

The SMILES alphabet has been defined based on the training set and the work of Arús-Pous et al. (2019). Similarly, the SELFIES alphabet has been developed using a library that converts SMILES to SELFIES (Krenn et al., 2020).

Since the alphabet has been extended to create a custom alphabet, containing approximately 200 symbols, checks have been included to ensure the validity of SELFIES strings. Regardless, SELFIES offer a more robust representation, as evidenced by results of the GAN models in Section 4.3.3.1. Finally, both these alphabets were used as sequential inputs for generative RNNs. As such, they required bespoke tokenisation based on the rules of the alphabet as described in Sections 4.2.1 (SELFIES) and 4.3.1.1 (SMILES).

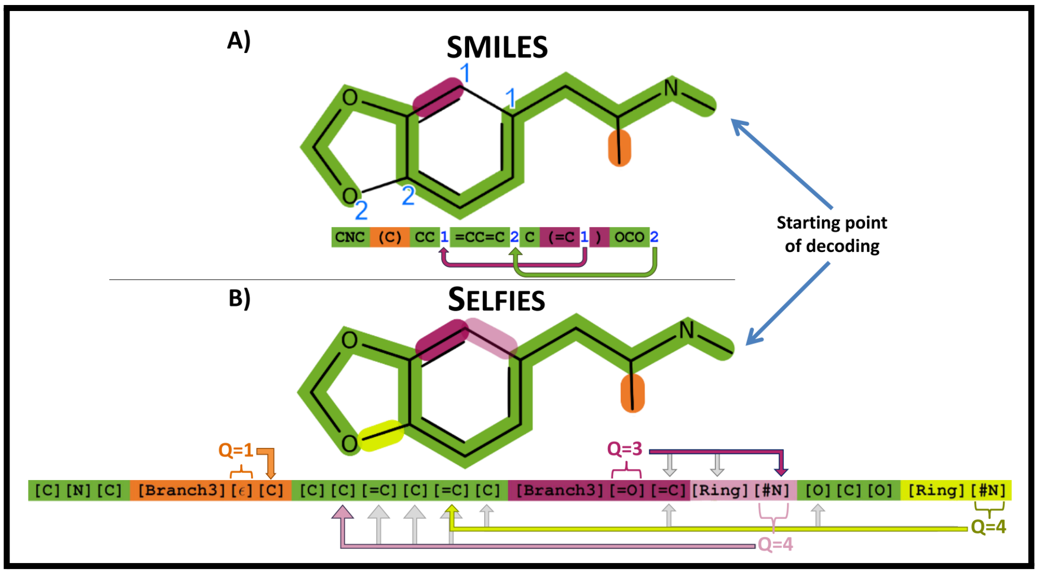

A)
SMILES
CNC (C) CC 1 =CC=C 2 C (=C 1) OCO 2
Starting point of decoding
B)
SELFIES
Q=1
Q=3
[C][N][C] [Branch3][ϵ][C] [C][C][=C][C][=C][C] [Branch3][=O][=C][Ring][#N] [O][C][O] [Ring][#N]
Q=4
Q=4

Figure 3.1 : (A) SMILES and (B) SELFIES representations of MDMA (Krenn et al., 2020)

3.3.2 Molecular Descriptors and Fingerprints

As described in Section 2.1.4, molecular fingerprints are the structures of molecules encoded as bit vectors. In this project, RDKFingerprints with a fixed length of 2048 bits are implemented through RDKit's Chem module. In addition to this, QSAR modelling techniques are employed using the *rdkit.Chem.Descriptors* and the *rdkit.Chem.QED* modules (The RDKit Documentation, 2022) to include the following descriptors as features:

a) **MolWt**: The average molecular weight of the molecule
b) **MolLogP**: The lipophilicity of the compound
c) **NumHDonors**: The number of hydrogen bond donors
d) **NumHAcceptors**: The number of hydrogen bond acceptors
e) **NumRotatableBonds**: The number of rotatable bonds in the molecule
f) **TPSA**: The topological polar surface area
g) **QED**: The quantitative estimate of drug likeness

3.3.3 Graphs

A graph is mathematically defined as a set of edges E and a set of vertices V. From this, one can intuitively understand atoms as the nodes or vertices of the graphs and the bonds between the atoms as the edges of a graph. To preprocess these abstract molecular graphs into features which can be handled by a computer, the sets V and E need to be mapped to linear data structures such as arrays or matrices. For this project, the Deep Graph Library as per Li et al., 2021, and RDKit were used to achieve this in order to train a graph neural network. The details for this preprocessing are discussed further in Section 4.1.2.

4. Design and Implementation

4.1 Classifier Model Implementations

To solve the generation problem described in Section 1, information must first be obtained from a high-accuracy classifier in order to have confidence that the molecules produced by a generative model exhibit the desired properties. To that endeavour, multiple binary classification models (E. coli/1 or not E. coli/0) have been developed and tested. From these, the most accurate model will be chosen – in conjunction with other evaluation techniques – to be used as part of an examiner network for the most successful generative model. The following sections describe these models, the input format they used, how the input was preprocessed, and finally, the results of these models and their applicability to the problem statement.

4.1.1 Datasets

The prepared dataset was subsequently split into three different sets for training and testing. Every model in this section used the following datasets for training and testing:

a) **Stratified Test Set**: Created using stratified sampling of 20% for of the four sources of the compiled dataset. This ensured an even distribution of the sources within the test set.
b) **Random Test Set**: From the remaining data in the compiled dataset, 20% was selected non-uniformly for this test set. The distribution of the sources within this set is likely uneven.
c) **Training Set**: All the remaining data within the compiled dataset.

4.1.2 Graphs

The GNN model takes molecular graph representation of SMILES as input. These molecular graphs contain 2D and 3D structural information of chemical compounds. The GNN model was implemented in PyTorch using the nn.GCN class. DGL graphs were used to train the model, which involved converting the SMILES string into a Mol object and obtaining its adjacency matrix with the aid of the RDKit library. Next, the Ogb library was employed to extract feature vectors from each atom. The network consists of two graph convolutional layers, and the feature nodes were averaged to create a 2D output tensor for each graph, which corresponded to the number of classes being classified. Finally, the model used an Adam optimizer with a learning rate of 0.01 and a cross-entropy loss function.

4.1.3 SMILES Strings

The models implemented in WEKA are specifically designed to take raw text data as input and are able to handle the inherent variability and complexity of text data for classification purposes. The related models are *ZeroR*, *NaiveBayesMultinomialText*, and *SGDText*. The obtained datasets were stored as CSV files and subsequently converted to Attribute-Relation File Format (ARFF) files to be used directly within the WEKA environment for classification tasks. All models were implemented using the GUI tool. Default hyperparameters were used for each model (WEKA, 2022).

4.1.4 Molecular Descriptors

The compiled dataset was mostly preprocessed using Pandas and RDKit libraries. Pandas was used for tasks such as collating the data into dataframes and for checking for and removing duplicates. RDKit was used for converting SMILES strings into RDKFingerprints and other molecular descriptors. This produced a dataset containing 2060 features comprising the bits of the RDKFingerprint, the descriptors identified in Section 3.2.2, and the estimations of the Methods of Action(MOA) using descriptor thresholds. The models described in this section all use RDKFingerprints as one of the inputs. However, only some make use of the other molecular descriptors included in the dataset.

4.1.4.1 MLP

The standard Scikit-Learn implementation of an *MLPClassifier* object was implemented. This model was trained using all the preprocessed features from the dataset. The implementation made use of all the default values of the *MLPClassifier*'s hyperparameters except for the following changes:

a) The number of nodes in the four hidden layers were 32, 16, 8, and 4 respectively
b) The activation function used was *tanh*.

4.1.4.2 XGBoost

Scikit-Learn framework is used to treat the XGBoost models as classifiers, specifically, using the XGBClassifier model for this classification problem. This model was trained using all the preprocessed features from the dataset. After grid search fine-tuning, the final implementation made use of all the default values of the XGBClassifier's hyperparameters except for the following changes:

a) The number of estimators was changed to 100
b) The maximum depth was changed to 8.

4.1.4.3 Random Forest

The standard Scikit-Learn implementation of a *RandomForestClassifier* object was implemented. This model was trained using all the preprocessed features from the dataset. The implementation made use of all the default values of the *RandomForestClassifier*'s hyperparameters except for the number of estimators used; this was changed to 200.

4.1.4.4 Recurrent Neural Network

The RNN model was implemented in PyTorch and trained using RDKFingerprints from the dataset. The network consists of one hidden layer, which includes an LSTM layer and a fully-connected layer. Specifically, the input size, hidden size, and output size are 2048, 128, and 1, respectively. In order to evaluate the model's performance, binary cross-entropy with logits loss is selected as the loss function using the *nn.BCEWithLogitsLoss* method. Furthermore, the *optim.Adam* method is used as the optimizer, and the learning rate is set at 0.001.

4.1.5 Results

Reliable results were ensured by performing 10-fold cross validation on the training set and subsequently testing the trained models on the two test sets. The results of the cross validation and testing of the top 3 models are displayed in tables 4.1, 4.2, and 4.3 (see Appendix C for more). The best performing models are highlighted.

Model	Data Format	Precision	Recall	F1-Score	Accuracy %
Random Forest	Descriptor	0.900	0.890	0.890	89.41
XGBoost	Descriptor	0.880	0.880	0.880	88.00
MLP	Descriptor	0.820	0.820	0.820	81.93

Table 4.1 : Top 3 10-fold cross validation results

Model	Data Format	Precision	Recall	F1-Score	Accuracy %
Random Forest	Descriptor	0.900	0.900	0.900	89.61
XGBoost	Descriptor	0.885	0.880	0.880	88.33
MLP	Descriptor	0.820	0.820	0.820	81.89

Table 4.2 : Test 1 – Random Test Set results

Model	Data Format	Precision	Recall	F1-Score	Accuracy %
Random Forest	Descriptor	0.900	0.900	0.900	89.57
XGBoost	Descriptor	0.880	0.880	0.880	88.23
MLP	Descriptor	0.830	0.830	0.830	82.51

Table 4.3 : Test 2 – Stratified Test Set results

4.1.6 Final Classifier Evaluation

Using cross fold validation ensured that the model did not overfit to the training data by improving its ability to generalise on unseen data (Pramoditha, 2021). Additionally, one of the test sets was stratified to avoid any bias by keeping the proportion of samples collected from each source balanced, while the other was sampled randomly to use for comparison. Finally, one can see that metrics remain consistent across the board for each model when scored by the three evaluation techniques, ensuring that the models have been adequately assessed.

From Tables 4.1 - 4.3, it is observed that the RF and XGBoost models display the highest performance. However, the RF comes out on top. Furthermore, the RF achieved an unbeatable training time of approximately 5 minutes. This makes the model ideal for use in an examiner network as it will be used in tandem with a generative model. One possible explanation for the RF superior performance could be its propensity for classifying tabular data (Borisov et al., 2022) and has been previously successful when using molecular fingerprints as an input (Ahn et al., 2022; Zoffman et al., 2019).

With an accuracy of over 89%, the RF model has helped meet one of the primary aims: to develop a high-accuracy classification model that will identify molecules with the desired antibacterial properties against strains of escherichia coli (E. coli). As discussed in the introduction to this section, solving the classification problem will aid in the endeavour to solve the generation problem. As such, the highest performing model, i.e. the RF, can now be used to further the developed generative models on two fronts:

a) The output of RF is a confidence score, which indicates how likely it believes a given molecular fingerprint/descriptor belongs to that of a molecule that exhibits antibiotic properties towards E. coli. The average of this score, across a sample, will be used as an indicator of the performance of the generative models.
b) This score is also to be used as part of the examiner network in conjunction with other evaluation techniques such as toxicity and synthetic accessibility screening in the Generative Examination Network (GEN) models.

4.2 Generative Model Implementations

This section describes designing several generative models aimed at generating canonical SMILES strings. These models are designed to take a seed sequence, which is essentially a random character selected from the action space (the alphabet), and use it to generate new synthetic SMILES. The main goal of these models is to produce high-quality SMILES strings that resemble the real data distribution. This is achieved by exploring various generative models that can transform a random noise vector into a sequence of characters, which can then be decoded into a SMILES string.

4.2.1 Conversion To SELFIES

In addition to the SMILES representation used by the GAN, both GANs and GENs use the SELFIES format, obtained by using the external SELFIES library to convert SMILES strings to SELFIES. These strings provide a more robust and unambiguous representation of the molecular structure. Each SELFIES string is tokenized into individual elements representing atoms, bonds, and chemical notations. An alphabet and vocabulary dictionary are constructed for mapping between character tokens and integer representations. The "." (mixture) token is added to the character set, as the SELFIES implementation does not account for these.The data used for training consists of 123,985 SELFIES strings.

4.2.2 In-Silico Screening

Two forms of drug screening were used. The first is through using eToxPred, to check for synthetic accessibility, and toxicity. This form of drug screening was incorporated into the examiner, and was also conducted on the networks without the examiner.

The other way was through molecular docking. In order to do this, the following steps were taken :

a) RDKit was used to convert the SMILES into PDBs.
b) After this, Price et al. (2014) techniques were used to convert the PDBs into a suitable format for docking. PDBs were converted into PDBQTs using OpenBabel. During this conversion process, OpenBabel was used to add partial charges and hydrogen atoms to the molecules. This is as “AutoDock Vina requires input of PDBQT files that extend the standard PDB format with partial charges and atom types” (Price et al., 2014).
c) These PDBQTs were then docked onto target proteins, using AutoDock-Vina. Two proteins were selected for docking, each relating to a method of action for E. coli. These proteins were the DNA-gyrase (1KZN) and RNA-polymerase for E. coli. With this protein selection, the inhibition of protein synthesis and inhibition of nucleic acid synthesis MOAs are screened for.

The PDBs for these two proteins were downloaded from RCSB Protein Data Bank. The same pre-processing of the PDBs for the proteins was applied, to convert them into PDBQT files. AutoDockTools was then used to determine the grid space for the docking. The grid space determines the boundary of the docking. For testing, the grid space was set so that it covered the whole protein structure. After all data was prepared, docking commenced, with the outputs for each docking being written to a log file. Next, an average affinity score was given for each ligase screened. This was calculated by taking the highest binding affinity that the ligase achieved on both proteins, and using the average of this across a sample of 100 generated molecules.

4.2.3 Metrics

Each metric is averaged over a generated batch. The metrics include:

a) **Average Validity**: This metric is an average of the following 3 validity scores:
   i. PDB Validity: % of SELFIES converted to valid PDB files for docking.
   ii. SMILES Validity: % of SELFIES converted to valid SMILES strings.
   iii. Molecule Validity: % of SELFIES converted to valid SMILES strings that could subsequently be converted to valid Mol objects.

b) **Uniqueness**: % of unique SELFIES generated (no duplicates).
c) **Novelty Train**: % of SELFIES not found in the training set.
d) **Novelty Ext**: % of SELFIES not found in external databases such as PubChem.
e) **Average Safety**: 1 subtracted by the toxicity score as predicted by EToX Model
f) **Average Synthetic Accessibility**: How easily the molecules of a batch can be synthesised where 0 being easily accessible and 1 being difficult. Scores were inverted to match the direction of other metrics.
g) **Overall Score**: Average of all of the above metrics.
h) **Average Binding Affinity**: Average.of highest binding affinity across 2 proteins
i) **Average RF confidence score**: The average of the Random Forest confidence score of all the molecules generated in a batch.

4.2.4 GAN using Recurrent Neural Networks

A conditional GAN (cGAN) architecture is used to enable the generation of specific classes of compounds based on input labels. The generator's role is to generate synthetic SMILES strings, while the discriminator's role is to distinguish between the real and the generated samples.

The generator is designed to take a noise vector, label information, and previous token information as input and generate a sequence of tokens representing a SMILES string. It comprises an initial concatenation layer that combines the noise and label information, followed by a dense layer and a *Reshape* layer to match the LSTM input shape. The LSTM layer captures the dependencies between tokens and generates the next token in the sequence. A time-distributed dense layer with *SoftMax* activation is used to generate probability distributions over the token vocabulary for each position in the output sequence.

The discriminator is designed to take a sequence of tokens representing a SMILES string and label information as input and classify the input as real or generated. It consists of an LSTM layer, followed by a concatenation layer that combines the LSTM output and the input label. Subsequent dense layers with *ReLU* activation are employed, leading to the final *Dense* layer with *Sigmoid* activation, which provides the probability of the input being real.

4.2.5 DRNN Generator

This model will make use of the SELFIES representation to generate new molecules. The SMILES strings can be converted into SELFIES and then tokenized and padded to a maximum length of 1544 tokens. The model will be trained on molecules which have antibacterial activity targeting E. coli. It will take as input an arbitrary input sequence and generate a sequence of tokens based on the seed and previously generated tokens. The model also makes use of an embedding layer which takes the inputted integer tokens and creates an embedding before being passed to the LSTM layers. The output layer produces a probability distribution over the characters in the SELFIES alphabet which can be used to sample the next token in the sequence.

It can then be extended to be made either conditional or contextual. A conditional generator has a secondary input to the model describing the class of the generated sequence. The secondary input can either be the class labels as defined by the dataset or the confidence score generated from the classification model. A contextual generator has the same architecture as the model above but the alphabet has an extra token, in this case '[ECOLI]', which can be prepended to all SELFIES strings that have antibacterial activity targeting E. coli. The idea behind this is to map the molecular space for all the molecules in the training set labelled as having antibacterial properties to this seed token. This can then be used to generate molecules

An examiner training environment is also proposed, where the generator could be combined with the classifier to improve the generated sequences' antibacterial properties. In this environment, the generator would be used to generate sequences that are scored by the classifier. Molecules above a certain threshold could be used to retrain the generator.

4.3 Preliminary Experiments for Generative Models

4.3.1 GANs

4.3.1.1 Preprocessing

The SMILES strings are tokenized into a list of tokens using a custom tokenizer function, which utilises a regular expression pattern to identify and separate tokens, capturing individual atoms, bond types, and special characters. The tokenized SMILES strings are one-hot encoded, and a character set is created to map between characters and indices. Start and end tokens are added to each tokenized SMILES string. Additionally, the SELFIES GAN uses tokenized and padded SELFIES strings that are then converted into integer representations using the character-to-integer mapping, enabling the model to process the data as numerical input.

4.3.1.2 Architecture

The GAN model consists of a generator and a discriminator, both of which use the *binary_crossentropy* loss function. The generator aims to maximise the likelihood of the discriminator classifying its output as real, while the discriminator aims to minimise the likelihood of misclassifying real and fake samples.

The generator is a conditional LSTM-based model that takes as input a latent noise vector (size: latent_dim), a label vector (size: label_dim), and a one-hot encoded previous token matrix (size: (max_len, num_tokens)). The noise and label are concatenated and passed through a *Dense* layer with

ReLU activation function, followed by a *Reshape* layer to match the LSTM input shape. An LSTM layer with 256 units and *return_sequences* set to True is followed by a *TimeDistributed Dense* layer with *SoftMax* activation function to produce the output probabilities for the generated tokens.

The discriminator is an LSTM-based model that takes as input the one-hot encoded tokenized sequences (size: (max_len, num_tokens)) and a label vector (size: label_dim). An LSTM layer with 256 units processes the input and the output is concatenated with the label. A series of *Dense* layers with *ReLU* activation functions is employed, followed by a final *Dense* layer with *Sigmoid* activation function to produce the classification output for real or generated samples.

4.3.1.3 Training

The GAN model is trained iteratively, involving separate training of the discriminator and generator. First, the discriminator is trained on a batch of sequences and their labels from the training data, as well as a batch which is randomly produced by the generator. The discriminator's loss is calculated as the binary cross-entropy between the predicted and true labels. Next, the generator is trained using the combined model, which consists of the generator followed by the discriminator with its trainable flag set to False. The input is a batch of noise vectors, labels, and previous tokens, and the output is the predicted validity score from the discriminator. The generator's loss is calculated as the binary cross-entropy between the predicted validity scores and true labels.

4.3.1.5 Generation

To generate new sequences, a batch of noise vectors and labels is passed to the generator, and the output is decoded into the corresponding string representation . The temperature parameter can be adjusted to control the diversity of the generated samples, with higher values leading to more diverse samples.

4.3.2 DRNNs

Four types of DRNNs were developed using SELFIES strings as follows:

a) ECOLI Only (EC) - Trained on only E. coli molecules
b) Conditional (CL) - Labels (0 or 1) as second input, indicating the class label
c) Conditional (Con.P)- RF confidence scores as secondary input
d) Contextual (CXT) - [ECOLI] token prepended to E. coli molecules

4.3.2.1 Preprocessing

a) Padding: To ensure a fixed input length, the tokenized SELFIES strings are padded with a special padding character. 3 different lengths (250,500 and 1544) were experimented with based on the distribution of the dataset.
b) Encoding: The tokenized and padded SELFIES strings are then converted into integer representations using a character-to-integer mapping. This enables the model to process the data as numerical input.

Percentile	**Tokenized sequence length**	**Inputs lost**
100	1545	0
99.99	643	12
99.95	484	62
99	213	1240
98	155	2480

Table 4.4: Input Length Statistics

c) Training data: The integer encodings are then separated into input and output sequences for the model where the input is one “timestep” or character behind the output sequence. Due to the size of the available data this was done in segments of 10000 molecules and saved to 13 pickle files to save RAM.
d) E. coli only models were filtered to only include tokens from SELFIES labelled as 1 from the training data halving the available data to learn from but also limiting the alphabet to characters found within previously synthesised target molecules.
e) Contextual models used an extra [ECOLI] token prepended to SELFIES labelled as 1.
f) Conditional models used labels as well as confidence scores obtained using the RF classifier.

4.3.2.2 Architecture

All models were iterated upon from a base generator. The DRNN generator consisted of three LSTM layers, each followed by a dropout layer, and a final *Dense* layer with a *SoftMax* activation function.

The input to the generator is a sequence of integers representing characters in the alphabet, with the length of the sequence as 1. This means the input is a single token, which will be used as a starting seed for generating a sequence. The *Embedding* layer is used to map each token to a dense vector of 256 dimensions, which allows the model to learn the relationships between tokens in the input sequence.

The LSTM layers are used to model the sequential dependencies between the characters in the input sequence, with each LSTM layer consisting of 256 LSTM cells. The ‘return_sequences=True’ argument in each LSTM layer means the layer returns the entire sequence of outputs, not just the last output.

The Dropout layers set at 0.2 are used to regularise the model and prevent overfitting by randomly dropping out a fraction of the input units during training. Finally, a Dense layer with a softmax activation function is used to predict the probability distribution over the alphabet of words for the next token in the sequence.

The model was compiled using the following hyperparameters:

a) Sparse Categorical Crossentropy loss function
b) Adam Optimizer with a learning rate of 0.0005 as generators are prone to overfitting.

Conditional models were extended with a secondary input for the class label or confidence score fed to an embedding layer before subsequently time distributed over a *Dense* layer the size of the latent dimensions to be concatenated to the generated sequence embeddings before being passed to the final *SoftMax* layer to obtain the probability distribution over the alphabet.

4.3.2.3 Training

The segments are shuffled and loaded one at a time to fit the model for 1 epoch with a batch size per segment of 1500 with running times of 2-4 hours. All models trained to an average loss of 0.2 and accuracy of 0.94.

4.3.2.4 Generation

The *generate sequence* function uses an autoregressive process which takes the previous tokens to get the probability distribution of the next token which is then smoothed using a temperature setting to then draw a multinomial distribution using '*np.random.multinomial*()' where a 1 is placed in the selected token. The index is obtained and used to select the next token from the alphabet. This is repeated until a set length. Given 0.05% or 62 molecules in the training set were above token length 484. A length of 500 was set to then compare the DRNN models. Through testing, it can be observed that increasing the temperature settings increases the diversity and creativity of the generated molecules however this comes with the an asterisk being that the validity and toxicity levels tend to increase.

4.3.3 Results

The best performing models for each network are highlighted. Cells highlighted in a different colour than the adjoining highlighted cells indicate the crucial metrics that were ultimately the deciding factors as to which model had the best performance.

4.3.3.1 GAN Results

Model / Input	**Avg Validity**	**Uniqueness**	**Novelty Train**	**Novelty Ext**	**Avg Safety (>=0.43)**	**Avg Syn Access (>=0.15)**	**Overall Score**	**Avg Binding Affinity**	**Avg RF confidence score**
SELFIES GAN	0.76	0.81	1.0	0.82	0.39	1.0	0.8	-5.69	0.51
SMILES GAN	0.43	0.28	1.0	0.29	0.36	0.70	0.59	-6.07	0.55

Table 4.5: GAN Metrics

4.3.3.2 DRNN Results

Generated molecules of length 500 tokens in batches of 100, using a temperature of 0.8. From this, one can observe that models that have seen both active and inactive molecules perform better and so the two best performers were chosen to be grid searched.

Model / Padding	Avg Validity	Uniqueness	Novelty Train	Novelty Ext	Avg Safety (>=0.43)	Avg Syn Access (>=0.15)	Overall Score	Binding Affinity	Avg RF confidence score
EC 500	0.87	0.90	1.0	0.70	0.22	0.99	0.80	-3.60	0.74
Con.P 500	0.97	1	1	0.73	0.3	0.97	0.86	-3.91	0.69
CL 1544	0.96	0.99	1.0	0.94	0.46	0.97	0.9	-3.77	0.53
CXT 500	0.95	0.99	1	0.85	0.38	0.94	0.88	-4.95	0.58

Table 4.6 : DRNN Metrics

The two best models from the initial comparison are then grid searched to choose a single model. This being the contextual model trained with padding 500, max generation length of 500 and temperature of 5.

Model / Temp / Output length	Avg Validity	Uniqueness	Novelty Train	Novelty Ext	Avg Safety (>=0.43)	Avg Syn Access (>=0.15)	Overall Score	Avg RF confidence score
CXT500 5 500	0.99	1	1	1	0.48	1	0.93	0.74
CXT500 1.5 250	0.99	1	1	0.88	0.38	0.98	0.91	0.69
CL1544 0.8 500	1	0.96	1	0.96	0.45	0.99	0.92	0.53
CL1544 0.8 250	0.99	0.92	1	0.96	0.44	0.96	0.90	0.58

Table 4.7: 2 Best DRNN models grid searched to get best model

4.3.4 Evaluation

The best-performing DRNN model was compared to the best-performing GAN, i.e. the SELFIES GAN. The SELFIES GAN showed promising results in generating molecules with high binding affinities, however, it was observed that the generated molecules from the GAN tend to be structurally similar as shown by the uniqueness scores and chemical graphs (see Appendix B). This indicates a lack of diversity in the generated molecules. It should also be noted that the average safety of the molecules is below the chosen threshold of 0.43. This shows that the DRNN models surpassed the

GAN in terms of validity and overall performance. The DRNN models were capable of generating molecules that fell within the defined safety range of greater than 0.43. Moreover, the best-performing classifier (RF) indicated an average confidence score of over 0.5 for the antibacterial properties of the generated models.

4.3.5 GENs

The DRNN models can then be further extended with examiner network which takes a pre-trained generator model and uses a combination of QSAR screening and classification models to filter the generated outputs of the generator based on validity, uniqueness, toxicity, synthetic accessibility (SynAccess/SA) and RF confidence scores. This combined network forms a GEN. The GEN is then trained for a set number of generations. The best performing DRNN model, i.e. the contextual model with a padding of 500, generation length of 500 and temperature of 5, was used as a starting point for the examiner network. Additionally, this model used the best hyperparameters found through the grid search as observed in Table 4.7.

The temperature parameter was increased proportional to the number of duplicate sequences in a generated batch and was down-regulated based on the number of SELFIES strings that successfully converted to SMILES strings then RDKIT Molecule objects and then fingerprints.

The random forest classifier used a fixed threshold of 0.5 to filter the molecules, with the molecules below this threshold being pruned. Let A be the current generation of the GEN, and B be the total number of generations the GEN is set to run for. X is equal to A/B. The toxicity and SA models used upsampling in order to filter the molecules, allowing only the top X% least toxic and top X% most synthetically available through. Let FAAB be the top X% least toxic molecules taken from the approved antibiotic dataset. To do this, the approved antibiotic dataset is examined by the toxicity network, in order to get the toxicity scores. Let N be the number of molecules that passed screening. The generator is retrained with the generated molecules that passed screening, and a random sample of size N from FAAB. Figure 4.7 illustrates this process.

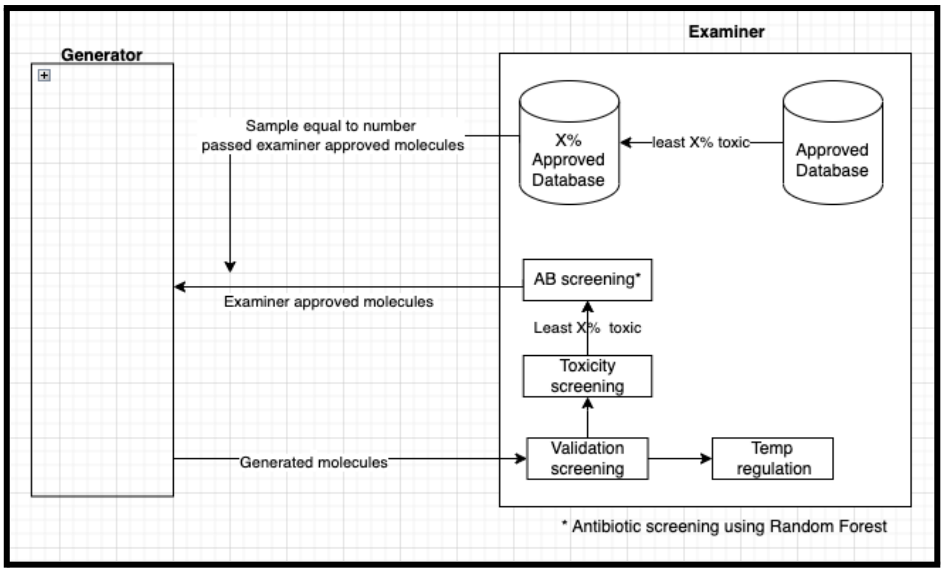


Figure 4.1 : Diagram illustrating the screening process in the GEN.

The grid search method was then used to find the optimal values for generation length and temperature in the GEN. Within the grid search, temperature values ranged from 0.8-8, with generation lengths 250 and 500 being trialled for each temperature setting. Table 4.8 contains the best results from the grid search.

Len	**Temp**	**Avg Validity**	**Uniqueness**	**Novelty Train**	**Novelty Ext**	**Avg Safety (>=0.43)**	**Avg Syn Access (>=0.15)**	**Overall Score**	**Avg RF confidence score**
250	3.5	1	1.0	1.0	0.96	0.41	0.99	0.93	0.74
	5	0.97	1.0	1.0	0.88	0.48	1	0.92	0.69
500	1	0.93	1	1	0.92	0.42	0.94	0.88	0.53
	1.5	0.96	1	1	1	0.57	1	0.93	0.48

Table 4.8 : Contextual GENs results where PAD=500 and Temperature = 5 length=250

5. Validation and Evaluation

5.1 Final Model Evaluation

The final model chosen was the GEN, built on the pre-trained contextual DRNN. It was trained for 20 generations with a batch size of 25 and initial temp of 5. A grid search was then done to fine tune to the final parameters.

This model provides a robust and flexible approach to generating novel molecules in the SELFIES and SMILES representations, with evidence that these generated molecules can also be directed towards desired characteristics. Despite all the contextual models being given a common token, i.e. [ECOLI], they generated molecules that achieved a unique score of 1, indicating the model's capability for producing diverse molecules. The average RF confidence score of the final model was 0.69. Therefore, on average, the generated molecules were predicted to display antibiotic properties by the RF model. However, when compared to the base contextual DRNN model, it is observed that the inclusion of the examiner network decreased the model's average RF scores. This indicates that the RF model is less confident about its prediction on the antibacterial properties of the generated molecules. The average binding affinity of GEN-generated molecules was also lower.

However, binding affinity was calculated using only two mechanisms, making the score not truly representative of the efficacy. Nonetheless, it is still useful as an indicator of how well these molecules could work as antibiotics. Even though it seems that the likelihood of the generated molecules being antibiotics is lower, when compared to the contextual DRNN, the GEN did achieve a much higher safety score. It is possible that, with more resources, the examiner could be tuned to better manage the tradeoff between the safety and the predicted antibiotic properties of the generated molecules.

5.2 Comparison of Metrics with Approved Antibiotics

Docking was performed on a small sample of molecules from the approved antibiotics dataset and results provided in Table 5.1. A sample of 100 molecules was used. The largest binding affinity across the two proteins for the molecule was recorded and used in order to find the distribution of the binding affinities.

Approved Antibiotics Thresholds	**Binding Affinity (kcal/mols)**
Average Affinity	-6.1
50th percentile	-7.1
75th percentile	-7.5
25th percentile	-6.6

Table 5.1: Distribution of Binding Affinity

The distributions of the Safety and Synthetic Accessibility on the approved antibiotic dataset were also calculated. However, this was calculated using the entire dataset, rather than a sample of 100.

Approved Antibiotics Thresholds	Safety thresholds	Synthetic Accessibility
Avg	0.43	0.85
Max	0.63	1
75	0.48	0.96
25	0.38	0.78
min	0.19	0.0

Table 5.2: Distribution of Safety Thresholds and Synthetic Accessibility of Set of Approved Drugs

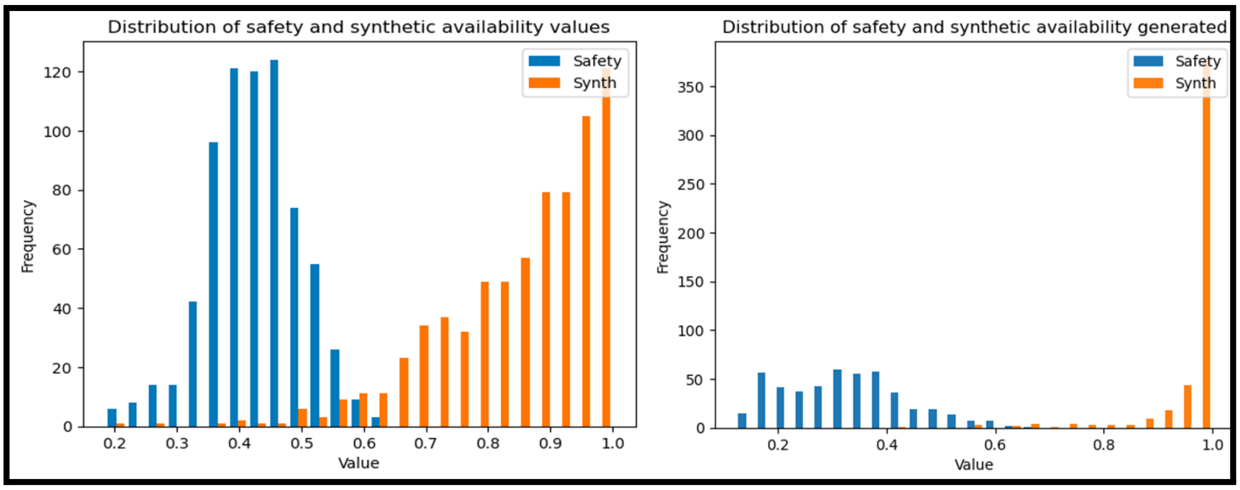


Figure 5.1 : Distribution of the safety and synthetic accessibility of approved drugs (left) and generated molecules (right)

It is observed that the average safety of the generated samples of the final model is 0.48, which would rank in the 75th percentile of approved drugs (0.48). This suggests that the generated samples would theoretically be safe for human use, according to the toxicity model. When comparing the SA, it is observed that the value of both the generated samples and approved drugs is 1. This suggests that the generated molecules could theoretically be synthesised with ease, according to the predictive model used. Such values can be attributed to the upsampling performed during the training of the GEN.

Finally, the generated molecules from the final model that scored lower than the average safety of the training set were discarded. Samples with a binding affinity ranking below the 25th percentile in the antibiotics dataset were also discarded. After this process, only three molecules from the generated batch of 100 remained. These molecules had binding affinities of -9.3, -6.9 and -6.8. See Appendix B for the chemical graphs for these molecules.

6. Conclusions and Future Work

6.1 Conclusions

In conclusion, this study demonstrates the effectiveness of DRNN models, particularly the contextual DRNN model enhanced with an examiner in generating novel molecules with predicted antibacterial activity against E. coli. The best-performing DRNN model exhibited superior validity and overall performance compared to the SELFIES GAN, generating diverse molecules with promising safety thresholds, synthetic accessibility, and average confidence scores of over 0.5 for antibacterial properties. Despite the GAN model's capability to produce molecules with high binding affinities, it suffered from a lack of structural diversity and lower safety scores. The examination network further enhanced the DRNN models, providing an efficient way to generate, filter, and evaluate novel molecules based on their validity, uniqueness, toxicity, synthetic accessibility, and target properties. With this final model, 100 molecules were generated. These molecules were then pruned according to their binding affinities and toxicities, resulting in three molecules, with a similar profile to approved drugs, being found.

Future improvements could involve increasing the batch size per generation, making the generation length variable, and padding, and incorporating additional drug characteristics into the QSAR analysis. Addressing these potential improvements would enhance the model's efficiency and reliability, contributing to the development of novel and effective antibacterial drugs. The generative models employed in this study provide a promising avenue for the discovery and development of novel antibacterial agents to combat drug-resistant E. coli infections.

6.2 Future Work

6.2.1 Incorporating binding affinities predictions into the examiner

As part of the evaluation process, molecular docking was done in order to determine how well the generated molecules would bind to key proteins. To extend the examiner, the classifier could be trained to predict these binding affinities in place of using docking tools such as Autodock Vina. This would allow us to incorporate this docking process into the generative model, and use it to better tailor drugs towards the desired antibiotic properties. While it is possible to incorporate the docking of generated molecules within training, docking can be time consuming, and it is possible that it would be faster for a model to predict these properties.

6.2.2 Predicting Drug Dosages

The ChEMBL database includes several Standard Activity types to characterise the type of bioactivity shown by a compound in a particular bioassay using standardised units to allow for a more unified measurement and comparison metric of the required concentration to achieve a given activity (ChEMBL; Papadatos et al., 2015). Some of these activity types include Minimum Inhibitory Concentration (MIC), Minimum Bactericidal Concentration (MBC), and IC50 (Half-Maximal Inhibitory Concentration). The IC50 is one of the most popular and informative measures of a drug's efficacy as it "indicates how much [of a] drug is needed to inhibit a biological process by half, thus providing a measure of potency of an antagonist drug in pharmacological research" (Aykul & Martinez-Hackert, 2016, p. 97). Similarly, the MIC and MBC of a compound indicates the lowest concentration of an antimicrobial that is required to visibly inhibit the growth of a microbe, which is specifically bacteria in the case of MBC. Several works have included these metrics as both features as well as targets for their models to improve their performance and increase their ability to predict inhibitory concentrations of compounds for a variety of pathogens, including E. coli (Joo et al., 2019; Mirzaei et al., 2021; Pataki et al., 2020; Yasir et al., 2022). As such, they offer a promising solution to improving the classification models by allowing them to predict not only the antibacterial activity of a given compound, but its dosage as well.

6.2.3 Further Screening for Modes of Action

The final model involved calculating the binding affinities of the generated molecules using the DNA-gyrase and the DNA-directed RNA-polymerase enzymes of E. coli cells – leading to the inhibition of nucleic acid and protein synthesis. To further enhance the model, one could incorporate further molecular docking by exploiting the other modes of action displayed by antibiotics. These include the:

a) inhibition of cell wall synthesis
b) disruption of cell membrane function
c) blockage of metabolic pathways.

To do so, one could acquire the relevant PDB files containing the binding receptors from a source such as the RCSB Protein Data Bank, and subsequently preprocess and dock the generated molecules as described in Section 4.2.2. For example, penicillin is a well-known antibiotic that targets cell wall synthesis (Etebu and Arikekpar, 2016) and as such, the penicillin-binding protein (PBP), PBP5, can be used as the docking receptor for the evaluation of a compound's affinity for binding with this protein. While the RCSB Protein Data Bank provides a comprehensive list of binding proteins, one should note that it is not exhaustive and sometimes lacks certain proteins as they may possess limitations, such as being too large to be used for molecular docking.

6.2.4 Drug Discovery as a Function-As-A-Service

The proposed generative model has shown promise in generating molecules, however, there is a notable drawback: a large variance in toxicity results from the generated molecules. This variance can cause outliers to be fed back into the model, affecting its reliability. One reason for the large variance in toxicity could be the low batch size of 100 used in the generation process. This small batch size is restricted by the limited RAM and CPU of standard machines. Therefore, employing cloud computing would facilitate further experimentation with generator hyperparameters, while also resolving the variance issue.

To implement this, a cloud provider such as Azure can be utilised to host two virtual machines. One machine serves as an examiner to evaluate the generated molecules as a Function as a Service (FaaS), while the other machine acts as the generator that takes hyperparameters to generate molecules. With the implementation of cloud computing, distributed grid searching can be performed to explore the effect of different hyperparameters on the variance of toxicity results, such as increasing batch sizes. This approach would solve the issue of limited computational resources, while also allowing further exploration of the generative model's capabilities.

6.2.5 Including toxicity and other QSAR metrics in the GAN

Incorporating additional data such as toxicity and other QSAR metrics into the GAN implementation could potentially enhance the model's performance and results. By conditioning the generator on these additional features, the generated molecules would not only have desired activity labels but also exhibit favourable physicochemical and pharmacokinetic properties, leading to a higher likelihood of being viable drug candidates. To achieve this, the input data can be expanded to include these QSAR metrics, and the generator and discriminator architectures can be modified accordingly to process and utilise this supplementary information.

Incorporating these additional QSAR metrics would require adapting the loss functions and evaluation metrics to account for the multi-objective optimisation problem. This can be achieved by introducing tailored loss functions that balance the trade-offs between various properties, such as activity, toxicity, and other desired molecular characteristics. Furthermore, the evaluation metrics should be extended to assess the quality of the generated molecules with respect to these additional properties, ensuring a comprehensive assessment of the model's ability to generate novel and biologically relevant compounds. This integration of QSAR metrics into the GAN framework is expected to result in a more robust model capable of generating diverse and drug-like molecules, thus accelerating the drug discovery process

7. References

1. Ahn, S., Lee, S. and Kim, M.-hyun (2022) “Random-Forest model for drug target-interaction prediction via Kullbeck–Leibler divergence.” Available at: https://doi.org/10.21203/rs.3.rs-1357588/v1.
2. Arús-Pous, J. et al. (2019) “Randomized smiles strings improve the quality of molecular generative models,” *Journal of Cheminformatics*, 11(1), p. 2. Available at: https://doi.org/10.1186/s13321-019-0393-0.
3. Aykul, S. and Martinez-Hackert, E. (2016) “Determination of half-maximal inhibitory concentration using biosensor-based protein interaction analysis,” *Analytical Biochemistry*, 508, pp. 97–103. Available at: https://doi.org/10.1016/j.ab.2016.06.025.
4. Baba, Y., Isomura, T. and Kashima, H. (2018) “Wisdom of crowds for Synthetic Accessibility Evaluation,” *Journal of Molecular Graphics and Modelling*, 80, p. 217. Available at: https://doi.org/10.1016/j.jmgm.2018.01.011.
5. Badura, A. *et al*. (2020) “Application of artificial neural networks to prediction of new substances with antimicrobial activity against *escherichia coli*”, *Journal of Applied Microbiology*, 130(1), pp. 40–49. Available at:https://doi.org/10.1111/jam.14763.
6. Benfield, A.H. and Henriques, S.T. (2020) “Mode-of-action of antimicrobial peptides: Membrane disruption vs. intracellular mechanisms,” *Frontiers in Medical Technology*, 2. Available at: https://doi.org/10.3389/fmedt.2020.610997.
7. Bento, C. (2021) *Multilayer Perceptron explained with a real-life example and python code: Sentiment Analysis*, *Medium*. Towards Data Science. Available at: https://towardsdatascience.com/multilayer-perceptron-explained-with-a-real-life-example-and-python-code-sentiment-analysis-cb408ee93141.
8. Bickerton, G., Paolini, G., Besnard, J. *et al.* (2012) “Quantifying the chemical beauty of drugs”, *Nature Chem,* 4, 90–98. Available at: https://doi.org/10.1038/nchem.1243
9. Bilodeau, C. *et al.* (2022) “Generative models for Molecular Discovery: Recent advances and challenges”, *WIREs Computational Molecular Science*, 12(5). Available at: https://doi.org/10.1002/wcms.1608.
10. Borisov, V. *et al.* (2022) *Deep neural networks and tabular data: A survey*, *Deep Neural Networks and Tabular Data: A Survey – arXiv Vanity*. Available at: https://www.arxiv-vanity.com/papers/2110.01889/ (Accessed: November 27, 2022).
11. Brownlee, J. (2019) *A tour of recurrent neural network algorithms for deep learning*, *MachineLearningMastery.com*. Available at: https://machinelearningmastery.com/recurrent-neural-network-algorithms-for-deep-learning/.
12. Brownlee, J. (2021) *How to develop a random forest ensemble in Python*, *MachineLearningMastery.com*. Available at: https://machinelearningmastery.com/random-forest-ensemble-in-python.
13. Bumgardner, B. *et al.* (2021) “Drug-Drug Interaction Prediction: A purely smiles based approach,” *2021 IEEE International Conference on Big Data (Big Data)* [Preprint]. Available at: https://doi.org/10.1109/bigdata52589.2021.9671766.
14. Capecchi, A., Probst, D. and Reymond, J.-L. (2020) “One molecular fingerprint to rule them all: Drugs, biomolecules, and the metabolome.” Available at: https://doi.org/10.26434/chemrxiv.11994630.

15. Chen, T. and Guestrin, C. (2016) “XGBoost,” *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* [Preprint]. Available at: https://doi.org/10.1145/2939672.2939785.
16. Coates, A.R.M., Halls, G. and Hu, Y. (2011) “Novel classes of antibiotics or more of the same?”, *British Journal of Pharmacology*, 163(1), pp. 184–194. Available at: https://doi.org/10.1111/j.1476-5381.2011.01250.x.
17. Corsello, S.M. *et al.* (2017) “The drug repurposing hub: A next-generation drug library and information resource,” *Nature Medicine*, 23(4), pp. 405–408. Available at: https://doi.org/10.1038/nm.4306.
18. Daina, A., Michielin, O. and Zoete, V. (2017) “SwissADME: A free web tool to evaluate pharmacokinetics, drug-likeness and medicinal chemistry friendliness of small molecules,” *Scientific Reports*, 7(1). Available at: https://doi.org/10.1038/srep42717.
19. Das, P. *et al.* (2021) “Accelerated antimicrobial discovery via deep generative models and molecular dynamics simulations”, *Nature Biomedical Engineering*, 5(6), pp. 613–623. Available at: https://doi.org/10.1038/s41551-021-00689-x.
20. David, L. *et al.* (2020) “Molecular representations in AI-Driven Drug Discovery: A review and practical guide” *Journal of Cheminformatics*, 12(1). Available at: https://doi.org/10.1186/s13321-020-00460-5
21. David, L. *et al.* (2021) “Artificial Intelligence and antibiotic discovery,” *Antibiotics*, 10(11), p. 1376. Available at:https://doi.org/10.3390/antibiotics10111376.
22. de Oliveira, E.C. *et al.* (2022) “Biological membrane-penetrating peptides: Computational prediction and applications,” *Frontiers in Cellular and Infection Microbiology*, 12. Available at: https://doi.org/10.3389/fcimb.2022.838259.
23. Dixit, V.A. (2018) “A simple model to solve a complex drug toxicity problem,” *Toxicology Research*, 8(2), pp. 157–171. Available at: https://doi.org/10.1039/c8tx00261d.
24. Ekins, S., Mestres, J. and Testa, B. (2007) “In silico pharmacology for drug discovery: Methods for virtual ligand screening and profiling,” *British Journal of Pharmacology*, 152(1), pp. 9–20. Available at: https://doi.org/10.1038/sj.bjp.0707305.
25. Epand, R.M. *et al.* (2016) “Molecular mechanisms of membrane targeting antibiotics,” *Biochimica et Biophysica Acta (BBA) - Biomembranes*, 1858(5), pp. 980–987. Available at: https://doi.org/10.1016/j.bbamem.2015.10.018.
26. Etebu, E. and Arikekpar, I. (2016) “Antibiotics: Classification and mechanisms of action with emphasis on molecular perspectives,” *International Journal of Applied Microbiology and Biotechnology Research*, p. 96. Available at: http://www.bluepenjournals.org/ijambr/pdf/2016/October/Etebu_and_Arikekpar.pdf (Accessed: January 2, 2023).
27. Fan, J., Fu, A. and Zhang, L. (2019) “Progress in molecular docking,” *Quantitative Biology*, 7(2), pp. 83–89. Available at: https://doi.org/10.1007/s40484-019-0172-y.
28. Feng, P., Wang, Z. and Yu, X. (2019) “Predicting antimicrobial peptides by using increment of diversity with quadratic discriminant analysis method,” *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 16(4), pp. 1309–1312. Available at: https://doi.org/10.1109/tcbb.2017.2669302.
29. Frank, E. *et al.* (2009) “Weka-A machine learning workbench for Data Mining,” *Data Mining and Knowledge Discovery Handbook*, pp. 1269–1277. Available at: https://doi.org/10.1007/978-0-387-09823-4_66.
30. Geurts, P., Ernst, D. and Wehenkel, L. (2006) “Extremely randomized trees,” *Machine Learning*, 63(1), p. 3. Available at: https://doi.org/10.1007/s10994-006-6226-1.

31. Goh, G.B. *et al.* (2018) *SMILES2VEC: An interpretable general-purpose deep neural network for predicting chemical properties*, *arXiv.org*. Cornell University. Available at: https://arxiv.org/abs/1712.02034.
32. Gómez-Bombarelli, R. *et al.* (2018) "Automatic chemical design using a data-driven continuous representation of molecules," *ACS Central Science*, 4(2), pp. 268–276. Available at: https://doi.org/10.1021/acscentsci.7b00572.
33. Goodfellow, I. *et al.* (2020) "Generative Adversarial Networks," *Communications of the ACM*, 63(11), pp. 139–144. Available at: https://doi.org/10.1145/3422622.
34. Grant, L.L. and Sit, C.S. (2021) "*De novo* molecular drug design benchmarking," *RSC Medicinal Chemistry*, 12(8), pp. 1273–1280. Available at: https://doi.org/10.1039/d1md00074h.
35. Guha, R. and Willighagen, E. (2012) "A survey of quantitative descriptions of molecular structure", *Current Topics in Medicinal Chemistry*, 12(18), pp. 1946–1956. Available at: https://doi.org/10.2174/156802612804910278.
36. Gupta, A. *et al.* (2017) "Generative recurrent networks for *de novo* drug design," *Molecular Informatics*, 37(1-2), p. 1700111. Available at: https://doi.org/10.1002/minf.201700111.
37. Hamouche, L., Poljak, L. and Carpousis, A.J. (2021) "Ribosomal RNA degradation induced by the bacterial RNA polymerase inhibitor rifampicin," *RNA*, 27(8), pp. 946–958. Available at: https://doi.org/10.1261/rna.078776.121.
38. Hochreiter, S. and Schmidhuber, J. (1997) "Long short-term memory," *Neural Computation*, 9(8), pp. 1735–1780. Available at: https://doi.org/10.1162/neco.1997.9.8.1735.
39. Ji, C. *et al.* (2018) "EMOLTOX: Prediction of molecular toxicity with confidence," *Bioinformatics*, 34(14), pp. 2508–2509. Available at: https://doi.org/10.1093/bioinformatics/bty135.
40. Joo, M. *et al.* (2019) "A deep learning model for cell growth inhibition IC50 prediction and its application for Gastric cancer patients," *International Journal of Molecular Sciences*, 20(24), p. 6276. Available at: https://doi.org/10.3390/ijms20246276.
41. Kadurin, A., Aliper, A., Kazennov, A., Mamoshina, P., Vanhaelen, Q., Khrabrov, K. and Zhavoronkov, A., 2017. The cornucopia of meaningful leads: Applying deep adversarial autoencoders for new molecule development in oncology. *Oncotarget*, *8*(7), p.10883. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5355231/
42. Kellis, M. and Jin, W. (2021) "AI for Drug Design", *Deep Learning in the Life Sciences*. Available at: https://www.youtube.com/watch?v=AHVJv5RNqKs&ab_channel=ManolisKellis (Accessed: November 20, 2022).
43. Kingma, D.P. and Welling, M., 2013. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*. Available at: https://arxiv.org/abs/1312.6114
44. Krenn, M. *et al*. (2020) "Self-referencing embedded strings (selfies): A 100% robust molecular string representation," Machine Learning: Science and Technology, 1(4). Available at: https://doi.org/10.1088/2632-2153/aba947.
45. Krenn, M. *et al.* (2022) "Selfies and the future of molecular string representations," *Patterns*, 3(10). Available at: https://www.sciencedirect.com/science/article/pii/S2666389922002069
46. Krenn, M. *et al.* (n.d.) *Aspuru-Guzik-group/selfies: Robust representation of semantically constrained graphs, in particular for molecules in chemistry*, *GitHub*. Available at: https://github.com/aspuru-guzik-group/selfies (Accessed: April 12, 2023).
47. Krogh, A. (2008) "What are artificial neural networks?," *Nature Biotechnology*, 26(2), pp. 195–197. Available at: https://doi.org/10.1038/nbt1386.

48. Kung-Hsiang, H. (2020) *A gentle introduction to graph neural networks (basics, DeepWalk, and GraphSage)*, *Medium*. Towards Data Science. Available at: https://towardsdatascience.com/a-gentle-introduction-to-graph-neural-network-basics-deepwalk-and-graphsage-db5d540d50b3.
49. Landrum, G. (2012) “Fingerprints in the RDKit,” *RDKit.org*. *RDKit 2012 UGM*, London. Available at:https://www.rdkit.org/UGM/2012/Landrum_RDKit_UGM.Fingerprints.Final.pptx.pdf (Accessed: April 18, 2023).
50. Li, J. *et al.* (2021) “Representative feature selection of molecular descriptors in QSAR modelling,” *Journal of Molecular Structure*, 1244, p. 131249. Available at: https://doi.org/10.1016/j.molstruc.2021.131249.
51. Li, M. *et al.* (2021) “DGL-LifeSci: An open-source toolkit for deep learning on Graphs in life science,” *ACS Omega*, 6(41), pp. 27233–27238. Available at: https://doi.org/10.1021/acsomega.1c04017.
52. Liu, Z. *et al.* (2021) “AI-based language models powering drug discovery and development,” *Drug Discovery Today*, 26(11), pp. 2593–2607. Available at: https://doi.org/10.1016/j.drudis.2021.06.009.
53. Livermore, D.M. (2004) “The need for new antibiotics”, *Clinical Microbiology and Infection*, 10(4), pp. 1–9. Available at: https://doi.org/10.1111/j.1465-0691.2004.1004.x.
54. Malvern Panalytical (n.d.) Binding affinity: Dissociation constant. Available at: https://www.malvernpanalytical.com/en/products/measurement-type/binding-affinity#:~:text=What%20is%20Binding%20Affinity%3F,(e.g.%20drug%20or%20inhibitor) (Accessed: April 25, 2023).
55. Mello, A. (2020) *XGBoost: Theory and practice*, *Medium*. Towards Data Science. Available at: https://towardsdatascience.com/xgboost-theory-and-practice-fb8912930ad6 (Accessed: November 20, 2022).
56. Mendez, D. *et al.* (2018) “Chembl: Towards direct deposition of bioassay data,” *Nucleic Acids Research*, 47(D1). Available at: https://doi.org/10.1093/nar/gky1075.
57. Mirzaei, M. *et al.* (2021) “A machine learning tool to predict the antibacterial capacity of nanoparticles,” *Nanomaterials*, 11(7), p. 1774. Available at: https://doi.org/10.3390/nano11071774.
58. *NaiveBayes Multinomial Text Class* (2022) . WEKA. Available at: https://weka.sourceforge.io/doc.dev/weka/classifiers/bayes/NaiveBayesMultinomialText.html (Accessed: March 28, 2023).
59. *NCI Dictionary of Cancer terms* (n.d.) *National Cancer Institute*. United States Government. Available at:https://www.cancer.gov/publications/dictionaries/cancer-terms/def/mechanism-of-action (Accessed: March 25, 2023).
60. Nouleho Ilemo, S. *et al.* (2019) “Improving graphs of cycles approach to structural similarity of molecules,” *PLOS ONE*, 14(12). Available at: https://doi.org/10.1371/journal.pone.0226680.
61. O'Boyle, N.M. *et al.* (2011) “Open babel: An open chemical toolbox,” *Journal of Cheminformatics*, 3(1). Available at: https://doi.org/10.1186/1758-2946-3-33.
62. Olivecrona, M. *et al.* (2017) “Molecular *de-novo* design through deep reinforcement learning,” *Journal of Cheminformatics*, 9(1). Available at: https://doi.org/10.1186/s13321-017-0235-x.

63. Papadatos, G. *et al.* (2015) "Activity, assay and Target Data Curation and quality in the CHEMBL database," *Journal of Computer-Aided Molecular Design*, 29(9), pp. 885–896. Available at: https://doi.org/10.1007/s10822-015-9860-5.
64. Pascanu, R., Mikolov, T. and Bengio, Y. (2013) *On the difficulty of training recurrent neural networks*, *PMLR*. PMLR. Available at: https://proceedings.mlr.press/v28/pascanu13 (Accessed: February 21, 2023).
65. Paszke, A. *et al.*, (2019). "PyTorch: An Imperative Style, High-Performance Deep Learning Library". *Advances in Neural Information Processing Systems 32. Curran Associates, Inc.*, pp. 8024–8035. Available at: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.
66. Pataki, B.Á. *et al.* (2020) "Understanding and predicting ciprofloxacin minimum inhibitory concentration in escherichia coli with machine learning," *Scientific Reports*, 10(1). Available at: https://doi.org/10.1038/s41598-020-71693-5.
67. Pedregosa, F. *et al.*, (2011). "Scikit-learn: Machine learning in Python". *Journal of machine learning research*, pp.2825–2830. Available at: https://arxiv.org/abs/1201.0490
68. Perron, Q. *et al.* (2022) "Deep generative models for ligand‐based *De Novo* Design applied to multi‐parametric optimization," *Journal of Computational Chemistry*, 43(10), pp. 692–703. Available at: https://doi.org/10.1002/jcc.26826.
69. Popova, M., Isayev, O. and Tropsha, A. (2018) "Deep Reinforcement Learning for de Novo Drug Design," *Science Advances*, 4(7). Available at: https://doi.org/10.1126/sciadv.aap7885.
70. Pramoditha, R. (2021) *How to mitigate overfitting with K-fold cross-validation*, *Medium*. Towards Data Science. Available at: https://towardsdatascience.com/how-to-mitigate-overfitting-with-k-fold-cross-validation-518947ed7428 (Accessed: April 28, 2023).
71. Prasanna, S. and Doerksen, R. (2009) "Topological polar surface area: A useful descriptor in 2D-QSAR," *Current Medicinal Chemistry*, 16(1), pp. 21–41. Available at: https://doi.org/10.2174/092986709787002817.
72. Price, G.W., Gould, P.S. and Marsh, A. (2014) "Use of freely available and open source tools for in silico screening in Chemical Biology," *Journal of Chemical Education*, 91(4), pp. 602–604. Available at: https://doi.org/10.1021/ed400302u.
73. Pu, L. *et al.* (2019) "EToxPred: A machine learning-based approach to estimate the toxicity of drug candidates," *BMC Pharmacology and Toxicology*, 20(1). Available at: https://doi.org/10.1186/s40360-018-0282-6.
74. Radu *et al.* (2022) *Convert FASTA TO CSV*, *Bioinformatics Answers*. Available at: https://www.biostars.org/p/3722/ (Accessed: December 17, 2022).
75. RDKit (2022) *rdkit.Chem.PandasTools module - The RDKit 2022.09.1 documentation*. RDKit. Available at: http://www.rdkit.org/docs/source/rdkit.Chem.PandasTools.html (Accessed: December 17, 2022).
76. *ReAct* (2021) "The world needs new antibiotics – so why aren't they developed?", 28 October. Available at: https://www.reactgroup.org/news-and-views/news-and-opinions/year-2021/the-world-needs-new-antibiotics-so-why-arent-they-developed/ (Accessed: November 25, 2022).
77. *REVIVE* (n.d.) "LogP". Available at: https://revive.gardp.org/resource/logp/?cf=encyclopaedia#:~:text=Definition%3A,an%20aqueous%20and%20lipophilic%20phase.

78. Roy, K., Kar, S. and Das, R.N. (2015) “Other related techniques”, *Understanding the Basics of QSAR for Applications in Pharmaceutical Sciences and Risk Assessment*, pp. 357–425. Available at: https://doi.org/10.1016/b978-0-12-801505-6.00010-7.
79. Sajed, T. *et al.* (2015) “ECMDB 2.0: A richer resource for understanding the biochemistry of E. coli,” *Nucleic Acids Research*, 44(D1). Available at: https://doi.org/10.1093/nar/gkv1060.
80. *SGDText Class* (2022) . WEKA. Available at: https://weka.sourceforge.io/doc.dev/weka/classifiers/functions/SGDText.html(Accessed: March 28, 2023).
81. *SMILES Tutorial* (2007) *EPA*. Environmental Protection Agency. Available at: https://archive.epa.gov/med/med_archive_03/web/html/smiles.html#:~:text=What%20is%20SMILES%3F,easily%20learned%20and%20flexible%20notation. (Accessed: March 29, 2023).
82. Stokes, J.M. *et al.* (2020) “A deep learning approach to antibiotic discovery”, *Cell*, 180(4). Available at: https://doi.org/10.1016/j.cell.2020.01.021.
83. Sutton, R.S., Bach, F. and Barto, A.G. (2018) *Reinforcement learning: An introduction*. Massachusetts: MIT Press Ltd.
84. *The RDKit book* (2022) *The RDKit Book - The RDKit 2022.09.1 documentation*. Available at: https://www.rdkit.org/docs/RDKit_Book.html (Accessed: March 25, 2023).
85. *The RDKit Documentation (2022) The RDKit 2022.09.1 documentation*. Available at: https://www.rdkit.org/docs/index.html (Accessed: March 25, 2023).
86. *Tokenizer Class* (2022). WEKA. Available at: https://weka.sourceforge.io/doc.dev/weka/core/tokenizers/Tokenizer.html(Accessed: March 28, 2023).
87. Trott, O. and Olson, A.J. (2009) “Autodock Vina: Improving the speed and accuracy of docking with a new scoring function, efficient optimization, and multithreading,” *Journal of Computational Chemistry* [Preprint]. Available at: https://doi.org/10.1002/jcc.21334
88. van Deursen, R. *et al.* (2020) “Gen: Highly efficient smiles explorer using autodidactic generative examination networks,” *Journal of Cheminformatics*, 12(1). Available at: https://doi.org/10.1186/s13321-020-00425-8.
89. Wishart, D.S. *et al.* (2017) “Drugbank 5.0: A major update to the DrugBank database for 2018,” *Nucleic Acids Research*, 46(D1). Available at: https://doi.org/10.1093/nar/gkx1037.
90. Wood, K.B. and Cluzel, P. (2012) “Trade-offs between drug toxicity and benefit in the multi-antibiotic resistance system underlie optimal growth of E. Coli,” *BMC Systems Biology*, 6(1), p. 48. Available at: https://doi.org/10.1186/1752-0509-6-48.
91. World Health Organization (WHO) (2020) *Antibiotic resistance*. Available at: https://www.who.int/news-room/fact-sheets/detail/antibiotic-resistance (Accessed: November 25, 2022).
92. Yasir, M. *et al.* (2022) “Prediction of antimicrobial minimal inhibitory concentrations for neisseria gonorrhoeae using machine learning models,” *Saudi Journal of Biological Sciences*, 29(5), pp. 3687–3693. Available at: https://doi.org/10.1016/j.sjbs.2022.02.047.
93. Zagidullin, B. *et al.* (2021) “Comparative analysis of molecular fingerprints in prediction of drug combination effects,” *Briefings in Bioinformatics*, 22(6). Available at: https://doi.org/10.1093/bib/bbab291.
94. *ZeroR Class* (2022) . WEKA. Available at: https://weka.sourceforge.io/doc.dev/weka/classifiers/rules/ZeroR.html(Accessed: March 28, 2023).
95. Zhavoronkov, A. *et al.* (2019) “Deep learning enables rapid identification of potent DDR1 kinase inhibitors,” *Nature Biotechnology*, 37(9), pp. 1038–1040. Available at: https://doi.org/10.1038/s41587-019-0224-x.

96. Zoffmann, S. *et al.* (2019) "Machine learning-powered antibiotics phenotypic drug discovery," *Scientific Reports*, 9(1). Available at: https://doi.org/10.1038/s41598-019-39387-9.

APPENDICES

APPENDIX A: Details of Dataset Curation

Alteryx Designer was used in the data preparation process of this project. Use of the software requires a licence. A free student/academic licence is obtainable.Use of the software requires a Windows operating system. More information about Alteryx Designer and its use is available here.

All datasets processed using the Alteryx Workflow Package. Source files and result files needed are provided in the 'datasets' folder of the project repository.

A.1 ChEMBL

Source: https://www.ebi.ac.uk/chembl/

I. Downloaded all ChEMBL SMILES as 'CHEMBL31_ALL_SMILES'.txt file.
II. Searched for 'escherichia coli' as a target.
III. Selected the activities of each of the six E. coli strains available.
IV. Downloaded the activities for each strain as .csv files and renamed them as '<strain>_ACTIVITIES'.
V. Used Alteryx Input Data Tool to input each csv file into an Alteryx workflow.
VI. Used Alteryx Select Tool to isolate the drug ID and SMILES fields of each .csv file.
VII. Used Alteryx Union Tool to unionise the six files.
VIII. Used Alteryx Unique Tool to remove duplicates and outputted the result as 'CHEMBL_ECOLI.xlsx' using Alteryx Output tool.
IX. Used Alteryx Join Tool to join the .txt file in (i) on the Right Anchor of the Join Tool with the result of (viii) on the Left Anchor of the Join Tool. SMILES were used as the Join Key. This isolated non-E. coli smiles in the resulting Right Unjoined output.
X. Sampled the first 59,994 records of the result of (ix) so as to have a balanced dataset.
XI. Used Alteryx Output Data Tool to output the result of (x) as 'CHEMBL_NO_ECOLI.xlsx'

A.2 Drugbank

Source: https://go.drugbank.com/releases/latest

A licence is required to obtain the Drugbank Database. Evidence of the academic licence acquired for this project is provided at the end of this appendix.

I. The full Drugbank database was downloaded as 'full_database.xml'.
II. The drug structures were downloaded as 'structures.sdf'.
III. The drug target identifiers were downloaded as 'all.csv'.
IV. Converted 'structures.sdf' to 'structures.csv' using 'sdf2csv.py'.
V. Used Alteryx Input Data Tool to input the files in (i), (ii) and (iv) into an Alteryx workflow
VI. Used Alteryx Select Tool to isolate the drug IDs of the records in (i).
VII. Used Alteryx Select Tool to isolate the drug IDs and SMILES of the records in 'structures.csv'.
VIII. Used Alteryx Filter Tool on all.csv to separate the file into E. coli and non-E. coli records.
IX. Used Alteryx TextToColumn Tool on both outputs of (viii) to get only the primary drug ID.

X. Used Alteryx Join Tool to join the full database to the E. coli output of (viii) using the drug ID as the Join Key.
XI. Used Alteryx Join Tool to join the full database to the non-E. coli output of (viii) using the drug ID as the Join Key.
XII. Used Alteryx Join Tool to join the output of (x) to 'structures.csv' using the drug ID as the Join Key.
XIII. Used Alteryx Join Tool to join the output of (xi) to 'structures.csv' using the drug ID as the Join Key.
XIV. Used Alteryx Select Tool to isolate the drug IDs and SMILES of the output of (xii).
XV. Used Alteryx Select Tool to isolate the drug IDs and SMILES of the output of (xiii).
XVI. Used Alteryx Unique Tool on the output of (xiv) to get unique SMILES.
XVII. Used Alteryx Unique Tool on the output of (xv) to get unique SMILES.
XVIII. Used Alteryx Output Data Tool to output the result of (xvi) as 'drugbank_ecoli.xlsx'.
XIX. Used Alteryx Sample Tool on the first 244 records of the output of (xvii) so as to have a balanced dataset.
XX. Used Alteryx Output Data Tool to output the result of (xix) as 'drugbank_no_ecoli.xlsx'.

A.3 ECMDB

Source: https://ecmdb.ca/downloads

I. Downloaded the entire database as 'ecmdb.json'
II. Converted the file in (i) to 'ecmdb.csv' using 'json2csv.py'
III. Used Alteryx Input Data Tool to input 'ecmdb.csv' into an Alteryx workflow.
IV. Used Alteryx Select Tool to select the drug ID and SMILES.
V. Used Alteryx Unique Tool to isolate unique SMILES.
VI. Used Alteryx Output Data Tool to output the result of (v) as 'ECMDB ECOLI.xlsx'

A.4 Repurposing Hub

Source: https://clue.io/repurposing

I. Downloaded the entire drugs database and samples database from the website as text files.
II. Searched for 'escherichia coli' on the website as a target and exported the file as a text file.
III. Used Alteryx Input Data Tool to input the files mentioned in I., II. and III.
IV. Used Alteryx Select Tool to isolate the ID and SMILES fields from the exported file in II.
V. Used Alteryx Output Data Tool to export the output of IV. as 'RH_ecoli.xlsx'.
VI. 'RH_ecoli.xlsx' had to then be manually cleaned due to more than one drug being in one row. The new file had 58 records and was named RH1_fixed.xlsx'.
VII. Used Alteryx Filter Tool on the drugs text file in I. to get launched drugs.
VIII. Used Alteryx Join Tool to join the output of VII. with the samples text file in I. with the 'pert_iname' field as the Join Key.
IX. Used Alteryx Unique Tool to remove duplicates.
X. Used Alteryx Select Tool to isolate the drug ID and SMILES.
XI. Used Alteryx Join Tool to join the output of IV.(Left Join Tool anchor) with the output of X. (Right Join Tool anchor) to isolate the Right Unjoined data (non-ecoli drugs).
XII. Used Alteryx Sample Tool to sample the first 58 rows of the output of XI. so as to have a balanced dataset.

XIII. Used Alteryx Output Data Tool to output the result of XII. as ‘RH0_fixed.xlsx’.

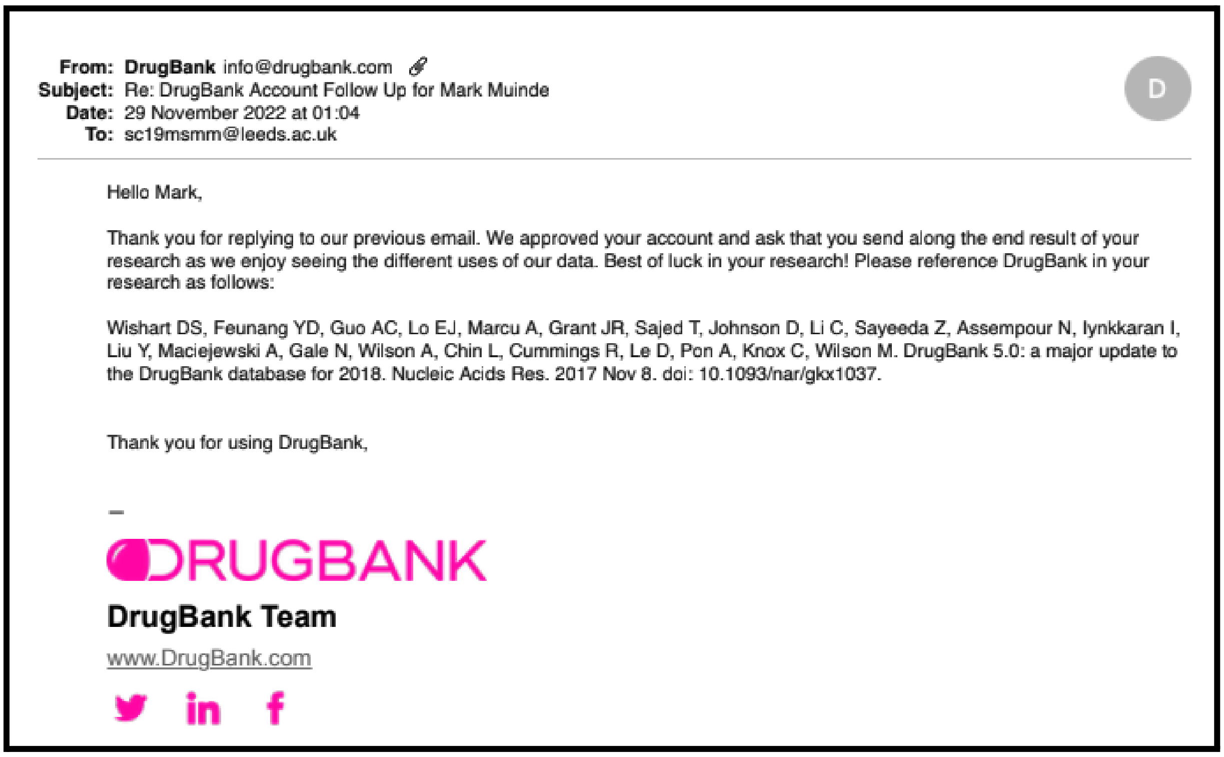
A.5 Full Dataset

After data preparation, each of the above-mentioned databases provided two datasets, one with E. coli drugs and one without.

[ChEMBL, Drugbank, ECMDB & Repurposing Hub were parsed further by the classifier and the resultant files that were used have the following naming convention: <dataset_source/ecoli status>_fixed.xlsx. For example, ‘ECMDB1_fixed.xlsx’ where ‘ECMDB’ is the dataset source, ‘1’ is the E. coli status, in this case this means the dataset contains drugs for E. coli whereas ‘0’ would mean non-E. coli drugs. For convenience, this naming convention was adopted for all other datasets.

I. The E. coli datasets were inputted into an Alteryx workflow using Alteryx Input Tool
II. The datasets in (i) were then unionised using Alteryx Union Tool.
III. Alteryx Formula Tool was used on the result of (ii) to add a one-hot-encoded field labelled ‘Ecoli Status’ and given the label ‘1’, indicating that it is an E. coli drug.
IV. The result of (iii) was the passed through Alteryx Unique Tool to remove duplicate SMILES.
V. The result of (iv) was then passed through Alteryx Data Cleansing Tool to remove null rows.
VI. The result of (v) was outputted as ‘combined_ecoli.xlsx’ using the Alteryx Output Data Tool.
VII. The non-E. coli datasets were inputted into an Alteryx workflow using Alteryx Input Tool
VIII. The datasets in (vii) were then unionised using Alteryx Union Tool.
IX. Alteryx Formula Tool was used on the result of (viii) to add a one-hot-encoded field labelled ‘Ecoli Status’ and given the label ‘0’, indicating that it is not an E. coli drug.
X. The result of (ix) was the passed through Alteryx Unique Tool to remove duplicate SMILES.
XI. The result of (x) was then passed through Alteryx Data Cleansing Tool to remove null rows.
XII. The result of (xi) was outputted as ‘combined_no_ecoli.xlsx’ using the Alteryx Output Data Tool.
XIII. The results of (iii) and (ix) were unionised using Alteryx Union Tool.
XIV. The results of (xiii) were passed through Alteryx Unique Tool to remove duplicate SMILES.
XV. The results of (xiv) were passed through Alteryx Data Cleansing Tool to remove null rows.
XVI. The result of (xv) was outputted as ‘combined_full.xlsx’ using the Alteryx Output Data Tool.

The following shows correspondence between a representative of the group and DrugBank, showing that DrugBank approved the use of their copyrighted data for this project:



From: **DrugBank** info@drugbank.com
Subject: Re: DrugBank Account Follow Up for Mark Muinde
Date: 29 November 2022 at 01:04
To: sc19msmm@leeds.ac.uk

Hello Mark,

Thank you for replying to our previous email. We approved your account and ask that you send along the end result of your research as we enjoy seeing the different uses of our data. Best of luck in your research! Please reference DrugBank in your research as follows:

Wishart DS, Feunang YD, Guo AC, Lo EJ, Marcu A, Grant JR, Sajed T, Johnson D, Li C, Sayeeda Z, Assempour N, Iynkkaran I, Liu Y, Maciejewski A, Gale N, Wilson A, Chin L, Cummings R, Le D, Pon A, Knox C, Wilson M. DrugBank 5.0: a major update to the DrugBank database for 2018. Nucleic Acids Res. 2017 Nov 8. doi: 10.1093/nar/gkx1037.

Thank you for using DrugBank,

–

DRUGBANK

DrugBank Team

www.DrugBank.com

Figure A.1: Approval from DrugBank

The full correspondence between DrugBank and a representative of the group is available in the GitLab repository wiki under ‘Supporting Material’.

APPENDIX B: Examples of Approved & Generated Molecules

B.1 SMILES GAN

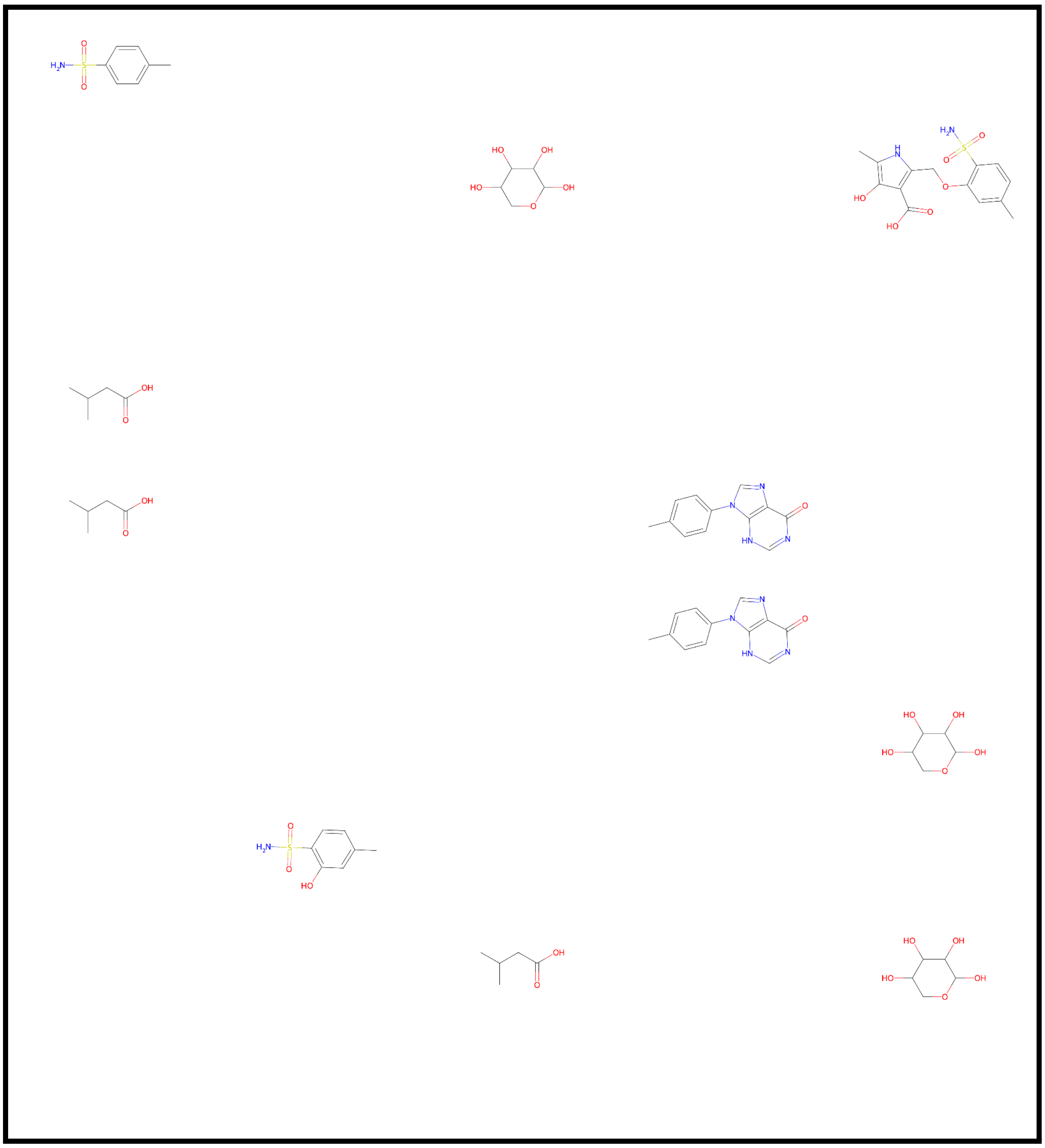


Figure B.1: Molecules generated by the SMILES GAN

B.2 SELFIES GAN

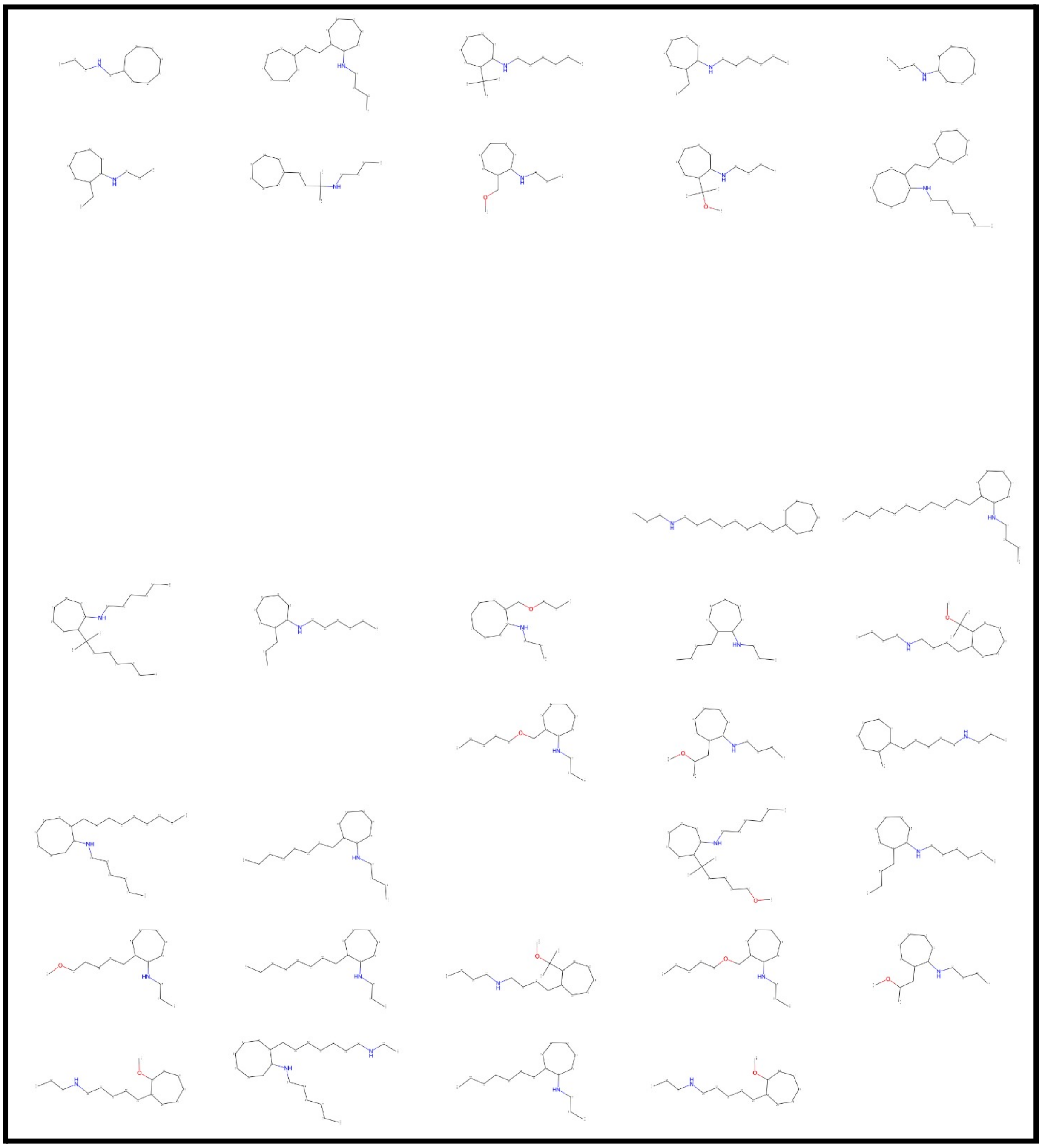


Figure B.2: Molecules generated by the SELFIES GAN

B.3 Best DRNN

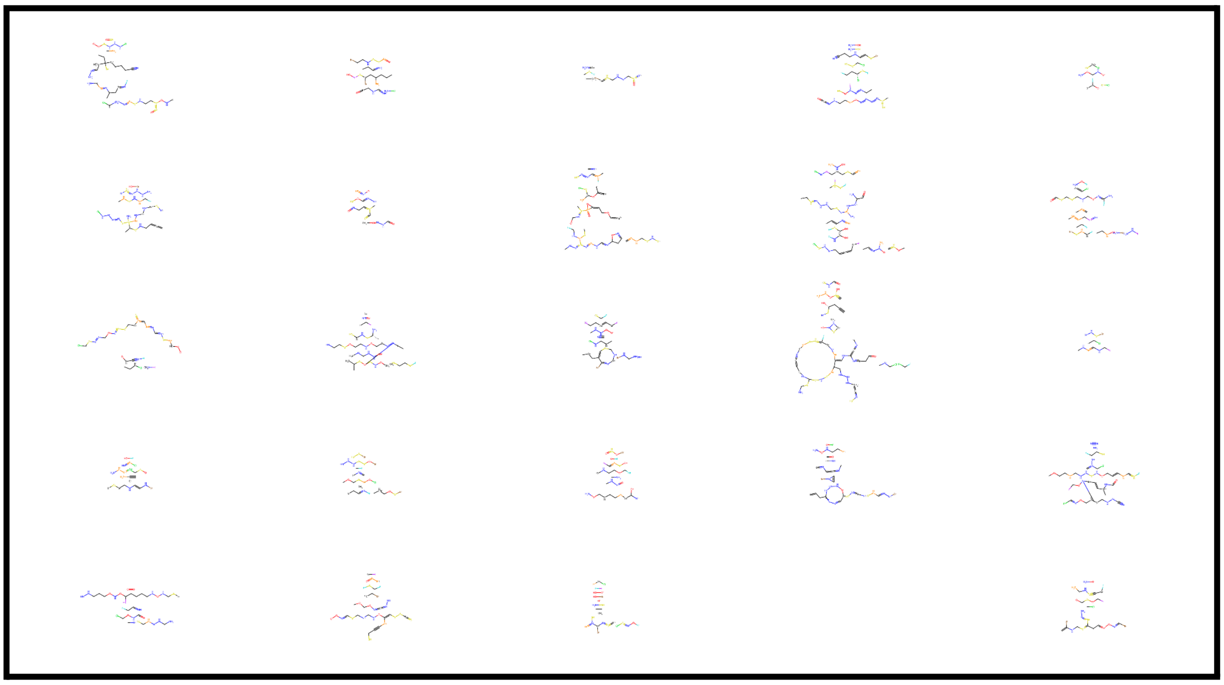
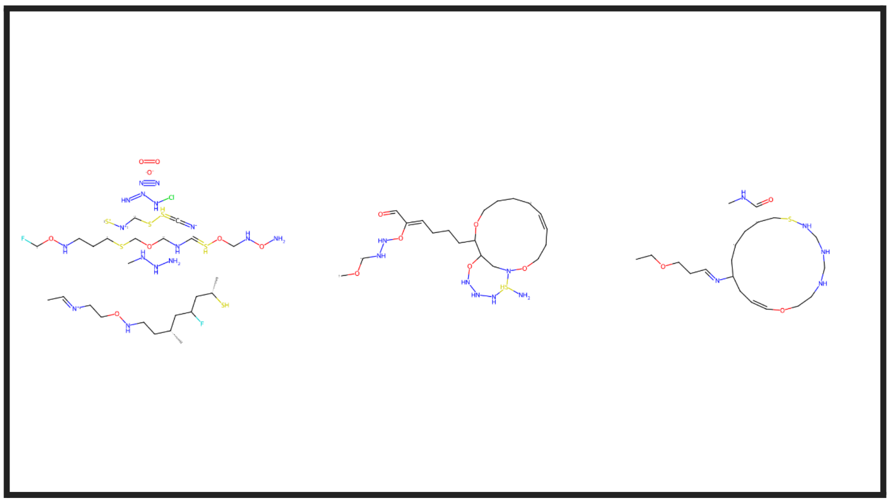


Figure B.3: Molecules generated by the Contextual DRNN using padding 500, temp 5, gen length 500.

B.4 Final GEN Model



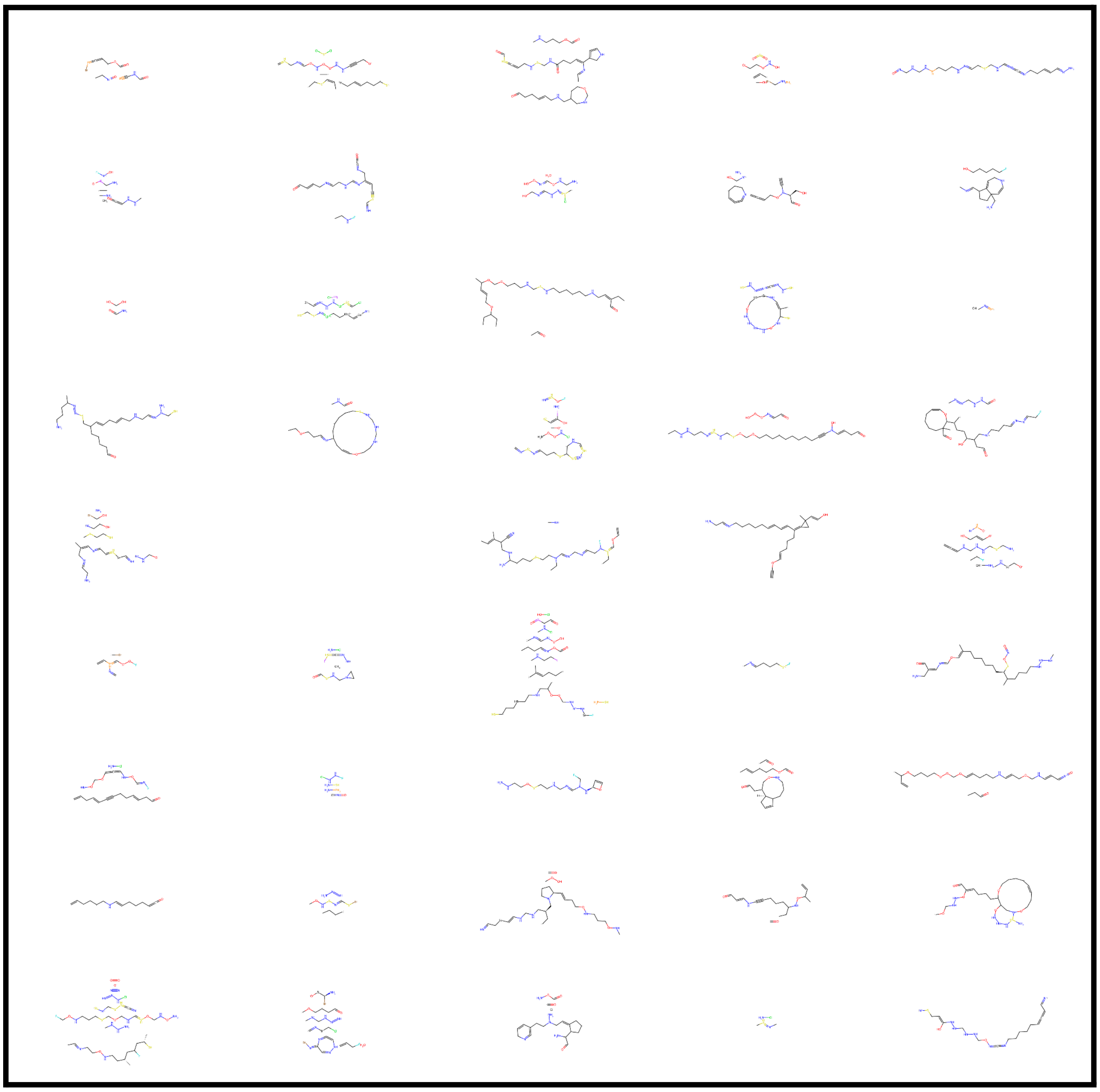


Figure B.4: Molecules generated by the Contextual GEN using padding 500, temp 5, gen length 250.

B.5 Approved Drugs

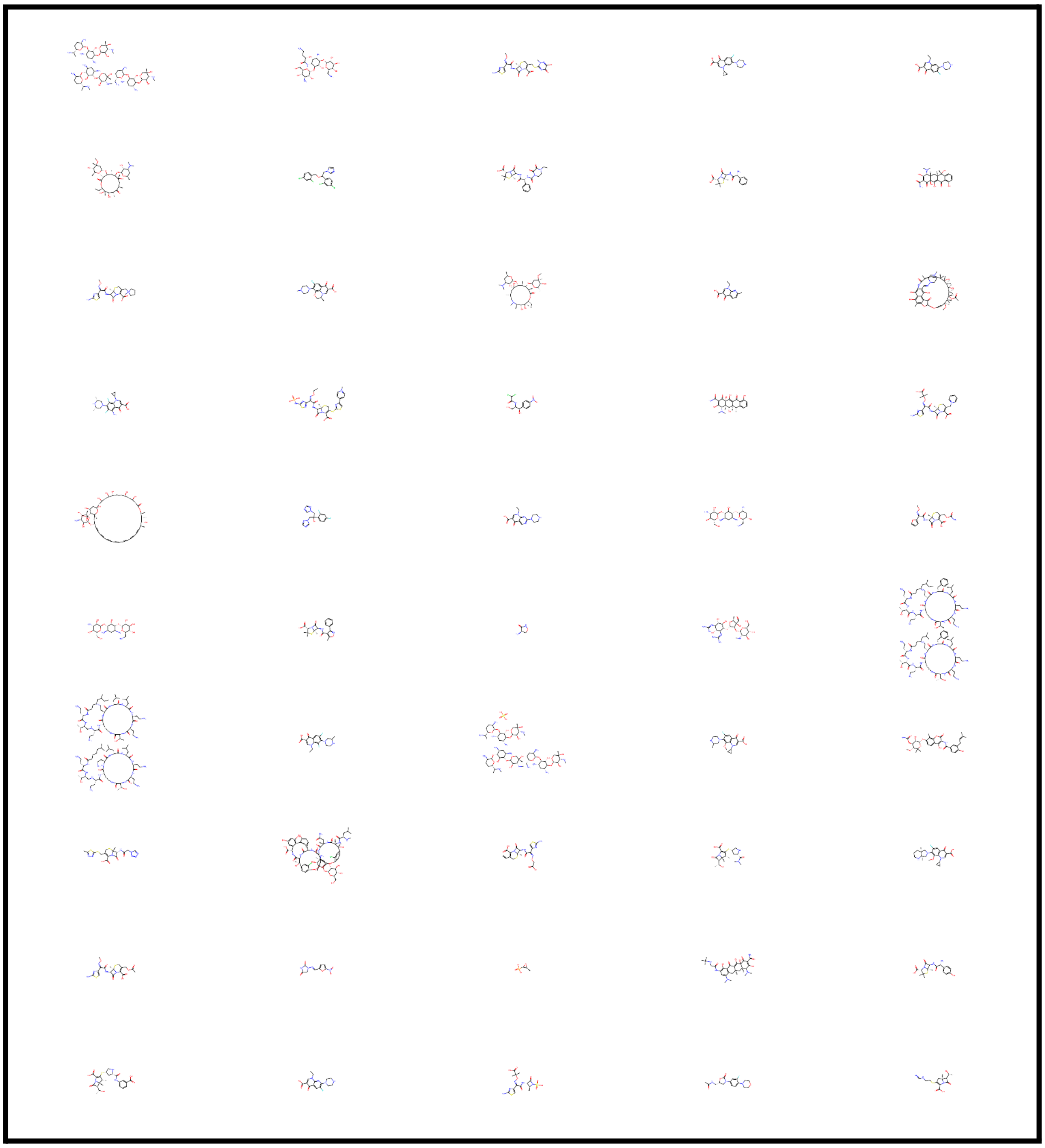


Figure B.5: Phase 4 tested (Approved) drugs obtained from ChEMBL.

APPENDIX C: Results of Classifier Experiments

C.1 Cross Validation

Model	**Data Format**	**Precision**	**Recall**	**F1-Score**	**Accuracy %**
Random Forest	Descriptor	0.900	0.890	0.890	89.41
MLP	Descriptor	0.820	0.820	0.820	81.93
Naive Bayes	SMILES	0.760	0.759	0.759	75.89
XGBoost	Descriptor	0.880	0.880	0.880	88.00
ZeroR	SMILES	0.500	0.500	0.500	50.00

Table C.1 : 10-fold cross validation results

C.2 Test 1

Model	**Data Format**	**Precision**	**Recall**	**F1-Score**	**Accuracy %**
Random Forest	Descriptor	0.900	0.900	0.900	89.61
MLP	Descriptor	0.820	0.820	0.820	81.89
Naive Bayes	SMILES	0.763	0.763	0.762	76.25
XGBoost	Descriptor	0.885	0.880	0.880	88.33
ZeroR	SMILES	0.505	0.505	0.505	50.52
SGD	SMILES	0.8667	0.717	0.7849	78.16

Table C.2 : Test 1 – Random Test Set results

C.3 Test 2

Model	**Data Format**	**Precision**	**Recall**	**F1-Score**	**Accuracy %**
Random Forest	Descriptor	0.900	0.900	0.900	89.57
RNN	Descriptor	0.807	0.837	0.821	81.79
MLP	Descriptor	0.830	0.830	0.830	82.51
Naive Bayes	SMILES	0.762	0.761	0.761	76.10
XGBoost	Descriptor	0.880	0.880	0.880	88.23
ZeroR	SMILES	0.501	0.501	0.501	50.10
SGD	SMILES	0.865	0.714	0.782	78.03

Table C.3 : Test 2 – Stratified Test Set results