Johannes Kepler University and University of South Bohemia in České Budějovice

Institute for Machine Learning

Bachelor's Thesis

# Generating and Classifying Molecules

*Author:*

Stefan Moser

*Supervisor:*

Univ.-Prof. Dr. Sepp Hochreiter

*Co-Supervisor:*

Dr. Mag. Günter Klambauer

Linz, 2019

## Bibliographical Detail

Moser, S., 2019: Generating and Classifying Molecules. Bachelor Thesis, in English. – 37 p., Institute for Machine Learning, Faculty of Engineering & Natural Sciences, Johannes Kepler University, Linz, Austria

## Annotation

Because in vitro drug design is resource intensive and therefore expensive, de novo drug design is the preferred approach. In this paper a data driven machine learning approach for generating and classifying molecules is presented. Using the data available via ChEMBL, a LSTM (Recurrent Neural Network) has been trained to generate molecules in SMILES format. 2.2 million molecules have been generated and screened for toxicity with an XGBoosted tree which has been trained on the Tox21 data set. This set contains 12 toxicity assays of the type nuclear receptor signaling (NR) and stress response (SR). In the end around 10,000 molecules have been classified as non-toxic and made available for public use.

## Declaration

I hereby declare that I have worked on my bachelor's thesis independently and used only the sources listed in the bibliography.

I hereby declare that, in accordance with Article 47b of Act No. 111/1998 in the valid wording, I agree with the publication of my bachelor thesis, in full to be kept in the Faculty of Science archive, in electronic form in publicly accessible part of the STAG database operated by the University of South Bohemia in České Budějovice accessible through its web pages.
Further, I agree to the electronic publication of the comments of my supervisor and thesis opponents and the record of the proceedings and results of the thesis defence in accordance with aforementioned Act No. 111/1998. I also agree to the comparison of the text of my thesis with the Theses.cz thesis database operated by the National Registry of University Theses and a plagiarism detection system.

. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

Date Stefan MOSER

# Contents

# Chapter 1

# Introduction

Chemistry is a field of science which influences peoples daily lives. Especially the invention of drugs is one that had a lasting impact on peoples lives.

But creating drugs is resource intensive. Estimations say that there are $10^{60}$[1] synthetically available drug-like molecules, but state of the art high throughput screening allows for only $10^6$ [2] molecules to be tested in the lab. This makes large in vitro experiments very cost and time intensive. This is where de novo drug design comes into play. Virtual screening allows for automatic creation and evaluation of molecules.

Currently, two different approaches exist for creating molecules. The first one builds novel molecules from predefined groups of atoms while the second one conducts virtual reactions based on rules set by experts. The problem with the first approach is that it artificially limits the possible molecules by using a fixed set. Segler et. al have recently shown that virtual reactions can sometimes fail[3].

The state of the art for classifying and filtering molecules are machine-learning approaches and docking. The machine-learning approaches can be divided into two categories: Target prediction, which labels molecules as either active or inactive, and Quantitative structure-activity relationships (QSAR) which predicts a real value (via regression) for the desired property of the substance. Popular descriptors are Structural Keys and Molecular Fingerprints. Neural networks and random forests are most commonly used for the prediction.

Inspired by the publication of Segler et al.[4] a pipeline for creating and labeling molecules with a data driven machine-learning approach has been implemented. The first part of the pipeline, the generation of valid compounds, is a recurrent neural network trained on a set of molecules. Given

one atom and its predecessors this model returns the probabilities of being the next atom for all possible atoms. A boosted tree which has been trained on the Tox21 data set is performing target prediction on the created molecules classifying them as toxic or non-toxic.
The target prediction inspects 12 different pathways (7 nuclear receptor signaling and 5 stress response assays) and if at least one of them is active the molecule is labeled as toxic.

# Chapter 2

# Methods

## 2.1 Data

### 2.1.1 ChEMBL

ChEMBL is a manually curated database which provides access to over 1.9 million bioactive drug-like small molecules. It is currently maintained by the European Bioinformatics Institute (EBI), of the European Molecular Biology Laboratory (EMBL) and part of the ELIXIR infrastructure. The data, which is abstracted and curated from primary scientific literature, covers a huge portion of structure–activity relationships (SAR) as well as newly discovered drugs. For each molecule ChEMBL stores the 2-D structure, calculated properties (like the Molecular Weight), and abstracted bioactivities. Its mission is to bring together chemical, bioactivity and genomic data to aid the translation of genomic information into effective new drugs[5].
Data can be acquired via File Transfer Protocol (FTP) or a web interface. For generating molecules, all SMILES from ChEMBL [6] have been downloaded via the web interface on 31.03.2019. This means out of 1.9 million available molecules more than 1.7 million (1,780,407 to be exact) serve as the data set for this task.

### 2.1.2 Tox21 Program

The Tox21 program was created by the US government and works together with the following federal agencies: the Environmental Protection Agency (EPA), the Food and Drug Administration

(FDA) as well as the National Institutes of Health (NIH). Leadership is provided by National Center for Advancing Translational Sciences (NCATS) and the National Toxicology Program (NTP) at the National Institute of Environmental Health Sciences. Each of these agencies brings an unique expertise to the organization[7]. The NTP provides animal toxicology knowledge while the EPA supplies computational toxicology knowledge. The FDA has human toxicity data and NCATS has in vitro cell-based assays, quantitative high-throughput screening and informatics proficiency.
The Tox21 program was founded because the current way of testing drugs is expensive, time-consuming and relies on animal testing. Therefore the goal of this endeavor is for the FDA and EPA to use the gathered data and created strategies on the products they regulate. To reach this goal three phases have to be cleared.
Phase 1 consisted of the creation of public data sets by analyzing 2,800 compounds in more than 50 assays. Phase 2 consisted of testing more than 10,000 compounds (Tox21 10K library) with an initial focus on stress response pathways and nuclear receptors. The resulting data set was also made publicly available. The third and final phase is comprised of updating the compound library, adding high-throughput gene expression profiling, training models via the created data sets and enlisting the help of the wider scientific community via crowdsourcing (which resulted in the Tox21 Challenge) and much more.

**Tox21 Data Challenge 2014:** was hosted by NCATS at NIH [8] and ran from August 18, 2014 to November 14, 2014. Its goal was to crowdsource data analysis to find out how well chemical properties can be predicted from their chemical structure alone. Winners were announced on January 12, 2015.
The provided data consisted of 12 assays of type nuclear receptor signaling (NR) for assay 1 - 7 and stress response (SR) for assay 8 - 12 (a visual representation can be found in Figure 2.1). They encompassed the following receptors for NR: estrogen receptor alpha, LBD (ER, LBD), estrogen receptor alpha, full (ER, full), aromatase, aryl hydrocarbon receptor (AhR), androgen receptor, full (AR, full), androgen receptor, LBD (AR, LBD) and peroxisome proliferator-activated receptor gamma (PPAR-gamma). For SR they gathered the following receptors: nuclear factor (erythroid-derived 2)-like 2/antioxidant responsive element (Nrf2/ARE), heat shock factor response element (HSE), ATAD5, mitochondrial membrane potential (MMP) and p53.
The challenge was divided into 15 subchallenges. Where subchallenges 1-12 consisted of predict-

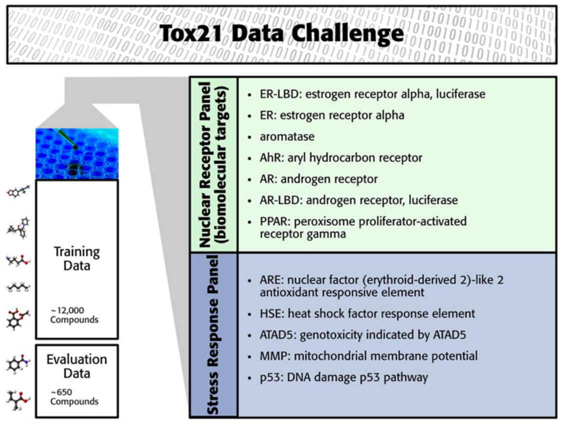

Tox21 Data Challenge
Training Data
~12,000 Compounds
Evaluation Data
~650 Compounds
Nuclear Receptor Panel (biomolecular targets)
ER-LBD: estrogen receptor alpha, luciferase
ER: estrogen receptor alpha
aromatase
AhR: aryl hydrocarbon receptor
AR: androgen receptor
AR-LBD: androgen receptor, luciferase
PPAR: peroxisome proliferator-activated receptor gamma
Stress Response Panel
ARE: nuclear factor (erythroid-derived 2)-like 2 antioxidant responsive element
HSE: heat shock factor response element
ATAD5: genotoxicity indicated by ATAD5
MMP: mitochondrial membrane potential
p53: DNA damage p53 pathway

Figure 2.1: Tox21 challenge overview [9]

ing individual assays, 13 and 14 were about a group of assays (NR and respectively SR) and the last subchallenge was the grand challenge where all assays were used.
All in all, three data sets were used in the competition. The first one was the training set which was available to the participants from the start and consisted of roughly 12,000 compounds. Next was a testing set consisting of nearly 300 molecules which was used to calculate the position on the leaderboard. The last one was the final evaluation data set which was used to determine the winner of the competition, it was compromised of around 650 compounds. The models had to predict a probability of a chemical being active in the assay as well as classifying it as active or inactive. The area under the receiver operating characteristic curve (area under the ROC curve) was used as evaluation criterion.

## 2.2 Data Representation

### 2.2.1 SMILES

Simplified molecular-input line-entry system (SMILES) is a standard for representing molecules in an (for humans) understandable way. It was initiated by David Weininger in the late 1980s and is currently managed by the Daylight Chemical Information Systems [10]. It also includes two other chemical languages: SMARTS and SMIRKS. If a SMILES string is canonicalized it serves as a unique identifier for the given compound.

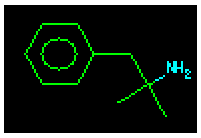


Figure 2.2: SMILES Phentermine: CC(C)(N)Cc1ccccc1

**Specification:** It represents molecules as ASCII strings. Where c and C are used for aromatic and aliphatic carbon atoms respectively. -, =, # and : represent single, double, triple and aromatic bonds. + and - can also represent a formal charge. Branches and cyclic structures can also be built with this notation by enclosing branches in parentheses and cycles in digits (illustrated in Figure 2.2) [11].

### 2.2.2 One Hot Encoding

One hot encoding is a technique for representing categorical data (in this case characters) as vectors of length K. With K being the number of possible values. All possible values are assigned an index i. To represent one value, a zero vector of length K is created and a 1 is assigned to the index i of the given value.
Given a set of possible values $\{C, H, O\}$ the corresponding one hot vector $x_t$ would be $x_t = (1, 0, 0)$ for C, $x_t = (0, 1, 0)$ for H and $x_t = (0, 0, 1)$ for O.

### 2.2.3 Structural Keys and Fingerprints

**Screens** are algorithms which detect the absence of a pattern in a molecule and are used for high-speed screenings of chemical databases. They look for the absence of patterns $(O(N))$ instead of their presence $\big(O(K^{kN})\big)$ because it is faster to detect the absence [12]. Screens are used to dismiss molecules which do not have the needed pattern with a 100% confidence, allowing the computationally more expensive substructure search to be used on the remaining molecules.

**Structural Keys** are bit vectors which mark specific patterns a molecule has. Which pattern each bit represents has to be decided beforehand. This is the cause of one of the problems structural keys have. Namely that their effectiveness is highly dependent on the chosen patterns. Another problem is that structural keys are normally very sparse, meaning only a few bits are active.
MACCS is a structural key that uses 166 bits, each corresponding to a SMARTS pattern. The

RDKIT implementation has been used in this paper [13].

**Fingerprints** are considered the next step in the evolution of screens. Each molecule has a unique fingerprint. To calculate it, one generates patterns representing each group of atoms and bonds connected by paths from 0 to bondlength bonds long. Using the molecule NC=CO as example the following patterns are generated:

- 0-bond paths: N, C, O
- 1-bond paths: NC, C=C, CO
- 2-bond paths: NC=C, C=CO
- 3-bond paths: NC=CO

This means all possible paths are generated. These patterns serve as seeds for the pseudo-random generation of bits representing them. As each pattern generates its set of bits, they are summed with a logical OR. Which means that if a molecule has a certain pattern these bits will definitely be set. This in turn allows one to easily dismiss molecules which do not have these particular bits set. It has to be mentioned though that fingerprints can not indicate with 100% accuracy that a pattern is present in a molecule if the pattern does not have a unique bit. If it does not have a unique bit all the other bits could have been set by a mix of patterns. Fingerprints are still superior to structural keys in most situations as they can store more patterns while still providing powerful screening capabilities.
A subclass of fingerprints, of which Morgan Fingerprints are a part of, are circular fingerprints. These do not look for specific features but rather use each heavy atom as a starting point for a descriptor. The amount of neighbors each descriptor examines is called radius.

## 2.3 Neural Networks

Neural networks are systems for machine learning that originated in the 1940s. They were created with the idea of mimicking the most powerful learner, the human brain.

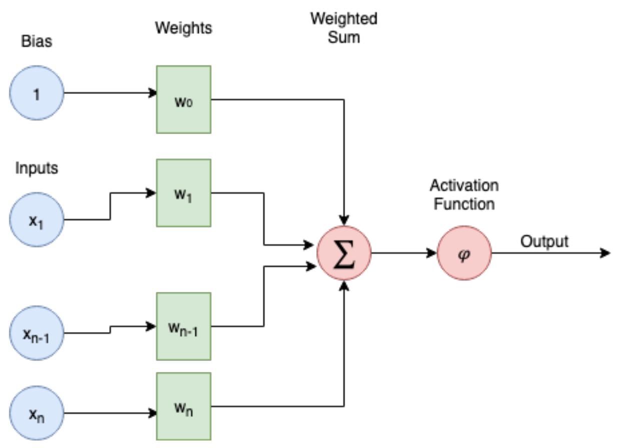


Figure 2.3: The layout of a single Perceptron

### 2.3.1 Multi-Layer Perceptrons

Multi-Layer Perceptrons are feed-forward artificial neural networks. They consist of perceptrons connected via weights over a variable amount of layers.
Their output is calculated via the so called forward propagation. It functions the following way: The input $x$ is passed to the first layer which then passes its output on to the next layer (multiplying it with the corresponding weight $w$) and so on until the output layer has been reached (compare Figure 2.4).
The output for one perceptron is calculated the following way:

$$g(x;w,\theta) = \varphi\left(\sum_{j=1}^{d} w_j * x_j - \theta\right)$$

where $g(x;w,\theta)$ is the classification function, $x$ is a matrix of inputs, $w$ holds the weights, $\theta$ the threshold and $\varphi$ is a differentiable activation function. (compare Figure 2.3)
The weights are the variables that can be adjusted to optimize the performance of the classifier. This is done using the back propagation algorithm.

**Back propagation:** Forward propagation is performed to calculate the difference between the computed and the desired output. Then from the last to the first layer the differences including the deltas from the previous layer are calculated.

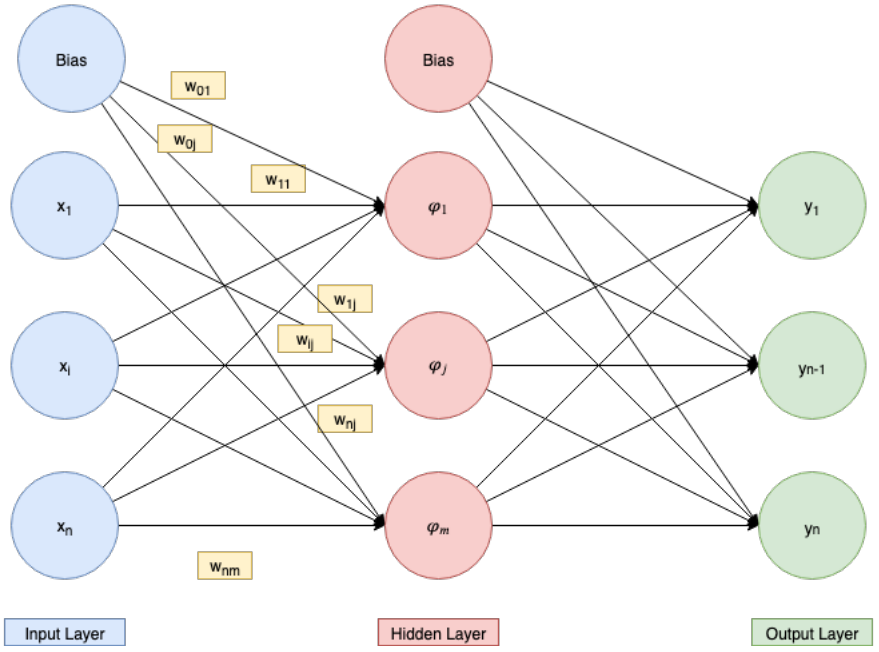


Figure 2.4: The layout of a Multi-Layer Perceptron

### 2.3.2 Gradient Descent

Gradient descent is the most popular way of optimizing neural networks. It minimizes the objective function $L(.;w)$ by adjusting the weights with its gradient and handles the intensity with the learning rate $\eta$. Due to its nature it is not guaranteed that the global minimum is found. Its formula is the following:

$$w_{n+1} = w_n - \eta \frac{\delta L(.;w)}{\delta w}$$

where $L$ is the loss function which is minimized, $\eta$ is the learning rate and $w_0$ are the initial values of the weights.

Three variants of gradient descent exist that differ in the amount of data used for one update.

**Batch Gradient Descent:** calculates the update for the whole training set. It is the most resource intensive because the whole data set has to fit into the memory but it converges to a local minimum or even a global one if the optimization landscape is convex.

$$w_{n+1} = w_n - \eta \frac{\delta L(x,y;w)}{\delta w}$$

**Stochastic Gradient Descent:** calculates the update for one sample at a time. It is faster than batch gradient descent, but its objective function fluctuates heavily due to the high amount of updates.

$$w_{n+1} = w_n - \eta \frac{\delta L(x^i, y^i; w)}{\delta w}$$

**Mini Batch Gradient Descent:** calculates the update for a subset of the training set. It gains the advantages of both methods in that it is faster and less resource intensive than batch gradient descent while not having a heavily fluctuating objective function. The only disadvantage is that the batch size $n$ is a hyper parameter that has to be optimized as well.

$$w_{n+1} = w_n - \eta \frac{\delta L(x^{i:i+n}, y^{i:i+n}; w)}{\delta w}$$

### 2.3.3 Deep Learning and Recurrent neural networks (RNNs)

Deep Neural Networks have recently emerged as one of the most powerful learners and are picked up by major tech companies around the globe. A neural network is deep when it has many hidden layers and broad when it has many hidden units.
It faces the problem of overfitting easily if not counteracted. Some of these measures include using an activation function that returns 0 a considerable amount of time (e.g.: Rectified linear units (ReLU) see Figure 2.5) or using a dropout rate. A dropout rate sets activations to 0 in proportion to its rate (e.g.: a dropout rate of 0.5 is a 50% chance of an activation being set to 0).
Another problem is that back propagation will not work properly with more than two to three layers because of vanishing gradients. The vanishing gradient problem is about updates converging to 0 for higher layers in gradient descent where the derivatives are 1.
Lastly, as neural networks take vectorial inputs and do not store prior inputs, they can not learn time series or sequences. One way of handling this would be by applying them to a sliding window. This method has the drawback of the net not being able to learn across windows due to the windows fixed size. To combat this problem recurrent neural networks (RNNs) have been invented.
For each forward pass over a window $t$ the activations are kept and potentially used in the next window $t + 1$ (compare Figure 2.6). This allows for a straight forward generalization of the forward pass. The generalization of the back propagation algorithm is called back propagation through time. Due to the activations being passed on between time steps the vanishing gradient problem is

amplified in RNNs. Limiting their learning abilities to around 10 time steps only.

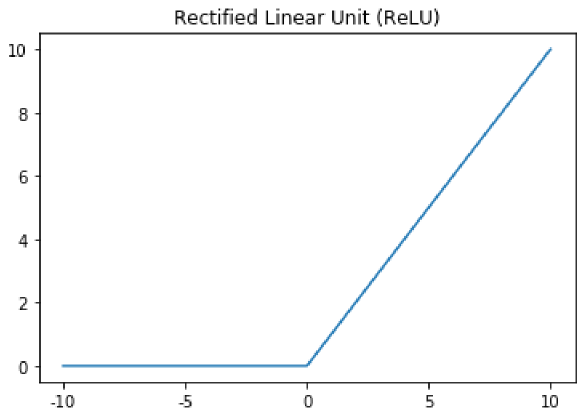


Figure 2.5: Rectified linear units activation function

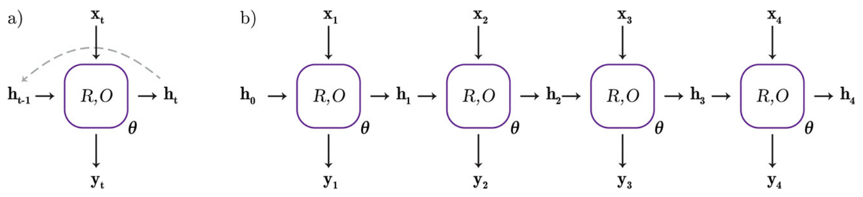


Figure 2.6: Scheme of an RNN workflow[4]

### 2.3.4 LSTM

A LSTM [14] is a special form of a Recurrent Neural Network (RNN) that enhances perceptrons to combat the vanishing gradient problem. It adds a linear self-connected memory unit (to combat the vanishing gradient), a multiplicative input gate (to combat irrelevant input), and a multiplicative output gate which only lets relevant content through (for a visualization see Figure 2.7).

Applied to the problem at hand, LSTMs keep track of time sequences which is crucial in the task of creating molecules. Given t letters they calculate the distribution of the t + 1 th letter. This means that the network predicts the most likely successors for a provided molecule. Then the chosen successor is handed to the network to predict the next one until a stop criterion is reached. (see Figure 2.6 for a visualization).

The number of possible characters for this network are all letters used in the SMILES data set downloaded from ChEMBL . Start and stop characters are also added to the dictionary because they enable the creation of a sequence by providing the start character as well as "automatic" stopping.

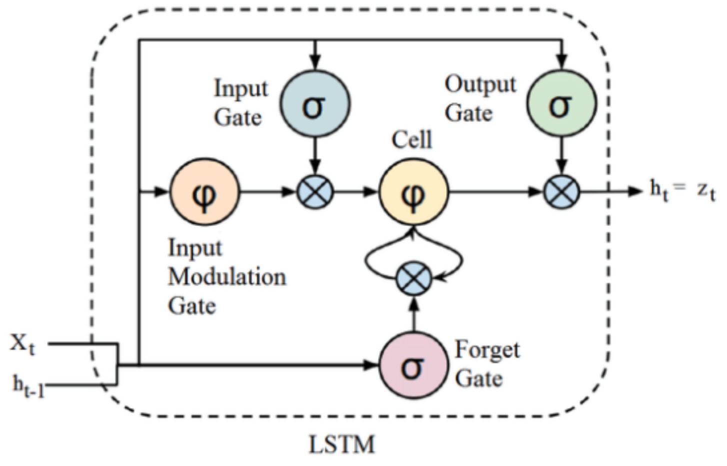


Figure 2.7: A LSTM cell [17]

For this implementation the Python [15] library Pytorch [16] and its implementations have been used.

## 2.4 Decision Trees

Decision trees are classifiers that explicitly describe which combination of features the prediction depends on. It starts at the root and then traverses the tree until it reaches a leaf which represents a class.
Such a tree encompasses three major design choices: which criterion to use to split, when to stop creating the tree, and whether to simplify the generated tree.
Splitting for categorical values can either be binary (is the value of type A) or on the entire feature (of which type is the value). To determine which option is the best, two types of splitting criteria exist.

**Information Gain:** uses entropy. For a variable $X$ with possible values $x_1, ..., x_n$ it is defined as:

$$H(X) = -\sum_{i=1}^{n} P(x_i)\ \log(P(x_i)) = \sum_{i=1}^{n} P(x_i)\, I(x_i) = E[I(X)]$$

Information gain compares how much information we can extract from a node by applying a specific split. If the data from the ith node Z is split into k splits the information gain is:

$$IG = H(Z,i) - \sum_{j=1}^{k} \frac{|Z_j|}{|Z|} H(Z_j)$$

**Gini Impurity:** uses the distribution of the labels to measure how often a random point would be mislabeled if it was assigned its class according to this distribution. It is defined as:

$$I_G(Z) = 1 - \sum_{j=1}^{J} p_j(Z)^2,$$

The gain then is defined as the amount of impurity we lose when applying a particular split:

$$g_G(Z) = I_G(Z) - \sum_{j=1}^{K} \frac{|Z_k|}{|Z|} \cdot I_G(Z_k)$$

**Stopping Criterion and Pruning Method:** are important for counteracting overfitting. Two pruning methods exist. The first one is reduced error pruning which removes all nodes not affecting the performance of the tree. The other one is called cost complexity pruning. It recursively removes the sub-tree which has the smallest error increase per leaf. Once only the root tree remains the tree which gives the best prediction performance is chosen as the final model.

### 2.4.1 Random Forest

Random forests are ensembles of decision trees. These trees are trained on a random subset of samples and features. This method is called bagging.

Due to random forests only using a subset of the training data they allow an out of bag estimate about the generalization performance. It is computed in the following way: For each tree one calculates the error with those training samples which have not been used to train the particular tree. Then the overall error is the mean of the individual errors.

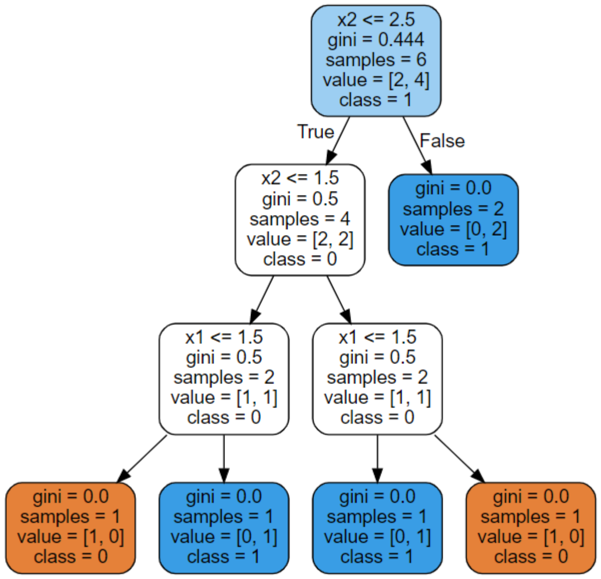


Figure 2.8: A sample decision tree

### 2.4.2 Gradient Boosting

Gradient Boosting is an ensemble technique that is used in supervised learning. Contrary to bagging where new models are created in parallel, new models are fitted sequentially on the residuals of the previous ones to correct the errors made by former models.

XGBoost stands for "Extreme Gradient Boosting" and is the next version of the algorithm. It fits the new model on the gradient instead of the residual to facilitate a greater variety of loss functions. Decision trees are the models that are optimized. They differ from random forests in the following way: while random forests create trees by subsampling features and data, XGBoosted trees successively add the tree that optimizes the gradient of the loss function of the previous model.

For this task the python library dmlc XGBoost (eXtreme Gradient Boosting) has been used.[18]

# Chapter 3

# Experiments

## 3.1 Molecule Generation

**Pre-Processing:** An initial analysis of the data set has shown that the distribution of SMILES lengths consists 97.64% of molecules that have a length lower than 150 characters (see Figure 3.1 and 3.2). To enable and optimize training on the available machines, all SMILES which are above said threshold have been dropped from the data set.
Afterwards the data set has been randomly split into train, validation, and test set with a ratio of 60:20:20.

**Training:** For generating molecules an LSTM which accepts a one hot vector of length equal to the possible number of molecules has been created. Its output is a vector of the same length as the input, consisting of the probability of being the next molecule for each possible molecule.

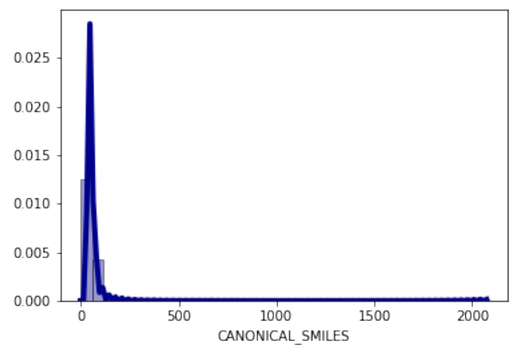


Figure 3.1: Distribution of molecule lengths before preprocessing

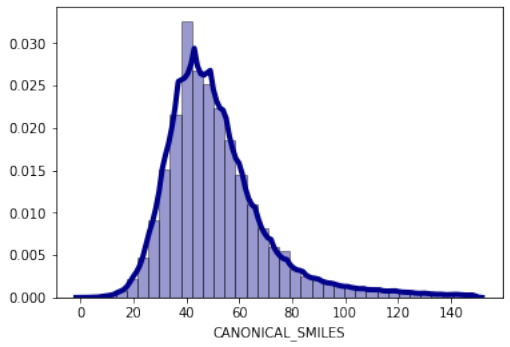


Figure 3.2: Distribution of molecule lengths after preprocessing

Its architecture is composed of 3 layers with 1024 hidden units each. In between layers a dropout of 0.2 has been utilized. To optimize cross entropy loss, mini batch gradient descent with a batch size of 128 and learning rate of 0.001 has been used. The model has been trained for 10 epochs and afterwards molecules have been generated from the epoch with the best performance on the validation set.

## 3.2 Molecule Classification

**Pre-Processing:** To address the imbalance in the Tox21 training data set the majority class has been undersampled before being split into a training and validation set with a 70:30 ratio. This has been done to reduce the rate of incorrectly classified toxic molecules. A test set has been provided by NCATS and allows comparison to the participants of the Tox21 challenge. To determine the best screen MACCS structural key and Morgan Fingerprints with different radii have been compared (see Table 3.1) using their respective ROC AUC. In the end the MACCS key outperformed all variants of Morgan Fingerprints and has therefore been chosen over the others. To further boost performance the following features have been added: variants of molecular weight (exact, average, without hydrogen), min/max (absolute) partial charge, Morgan Fingerprints with density 1 - 3 and the number of radical and valence electrons.

**Training:** For each of the 12 categories a xgboosted tree has been trained using the respective training set. Each boosted tree had a learning rate of 0.01 and minimum child weight of 5. Each tree in a boosted tree subsampled 80% of the available data and was allowed to grow to a max depth of 10. At each node 10% of the columns have been dropped. The ROC AUC has been used as evaluation criterion as it has also been used in the Tox21 competition and therefore enables a comparison to the participants of the challenge.

| | MACCS | Radius 1 | Radius 2 | Radius 3 | Radius 4 | Radius 5 | Radius 6 |
|---|---|---|---|---|---|---|---|
| nr-ahr | 91.27 ± 0.05 | 90.07 ± 0.66 | 90.25 ± 0.56 | 89.79 ± 0.43 | 89.88 ± 0.62 | 89.88 ± 0.38 | 89.84 ± 0.37 |
| nr-ar | 82.94 ± 0.42 | 83.78 ± 0.17 | 83.65 ± 0.13 | 83.32 ± 0.9 | 83.54 ± 0.28 | 82.64 ± 1.15 | 82.81 ± 0.85 |
| nr-ar-lbd | 87.19 ± 0.27 | 86.47 ± 0.43 | 86.55 ± 1.09 | 86.05 ± 0.26 | 86.16 ± 0.25 | 86.13 ± 0.41 | 86.32 ± 0.57 |
| nr-aromatase | 86.69 ± 1.39 | 85.38 ± 0.9 | 85.91 ± 0.27 | 86.20 ± 0.28 | 86.28 ± 0.12 | 86.47 ± 0.13 | 86.60 ± 0.17 |
| nr-er | 77.44 ± 0.16 | 77.18 ± 0.66 | 77.49 ± 0.52 | 77.40 ± 0.22 | 77.57 ± 0.2 | 77.50 ± 0.24 | 77.63 ± 0.13 |
| nr-er-lbd | 84.47 ± 0.64 | 83.95 ± 0.22 | 83.57 ± 0.58 | 84.07 ± 0.51 | 84.13 ± 0.24 | 83.94 ± 0.49 | 84.55 ± 0.15 |
| nr-ppar-gamma | 80.78 ± 2.47 | 79.50 ± 3.64 | 79.92 ± 2.9 | 79.63 ± 2.61 | 79.79 ± 2.55 | 79.88 ± 2.11 | 79.98 ± 1.74 |
| sr-are | 84.17 ± 1.09 | 81.85 ± 2.07 | 81.90 ± 1.78 | 81.81 ± 2.42 | 82.38 ± 2.03 | 82.00 ± 2.23 | 82.09 ± 1.96 |
| sr-atad5 | 85.59 ± 0.37 | 82.49 ± 0.48 | 82.42 ± 0.13 | 82.50 ± 0.28 | 82.98 ± 0.28 | 82.42 ± 0.16 | 82.52 ± 0.26 |
| sr-hse | 81.97 ± 2.83 | 80.75 ± 2.31 | 80.75 ± 2.09 | 80.85 ± 2.53 | 80.44 ± 2.87 | 80.50 ± 3.65 | 80.28 ± 3.79 |
| sr-mmp | 93.27 ± 0.42 | 91.05 ± 1.05 | 91.11 ± 0.89 | 91.14 ± 0.98 | 91.08 ± 1.21 | 91.23 ± 0.97 | 91.20 ± 1.08 |
| sr-p53 | 85.12 ± 0.59 | 81.59 ± 0.6 | 82.14 ± 0.14 | 82.38 ± 0.07 | 82.43 ± 0.21 | 82.14 ± 0.31 | 82.21 ± 0.31 |
| Average Score | 85.07 ± 0.63 | 83.67 ± 0.98 | 83.81 ± 0.8 | 83.76 ± 0.75 | 83.89 ± 0.81 | 83.73 ± 0.85 | 83.84 ± 0.82 |

Table 3.1: The ROC AUC scores (in %) for all tested screens

# Chapter 4

# Results

## 4.1 Generated Molecules

The generator reached its optimum at epoch 6 with a validation loss of 0.5123 and a cross-entropy loss of 0.5117 on the test set (as illustrated in Figure 4.1).

This model has been used to generate 2,500,002 molecules, 61,694 or 2.46%, of which could not be converted into RDKit molecules and were therefore deemed invalid. Another 192,813, 7.9% of valid molecules, were duplicates. After sorting out invalid molecules and duplicates 2,245,495 molecules, 89.82% of the original batch, remained (as illustrated in Figure 4.2). Even though only molecules of lengths shorter than 150 were used for training, molecules of lengths up to 300 have been produced (compare Figure 4.3).

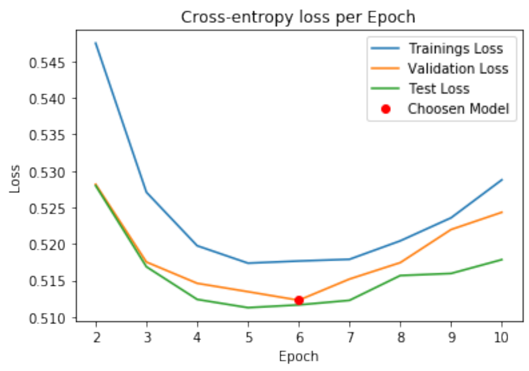


Figure 4.1: Train, Validation and Test loss

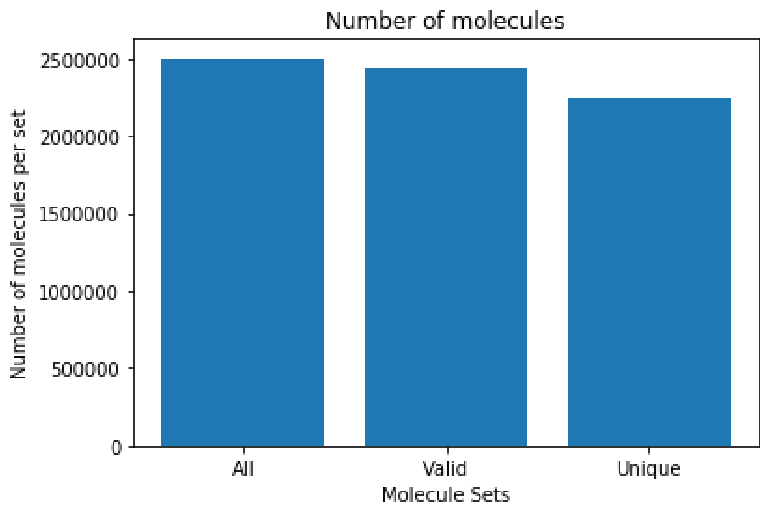


Figure 4.2: Molecules per set

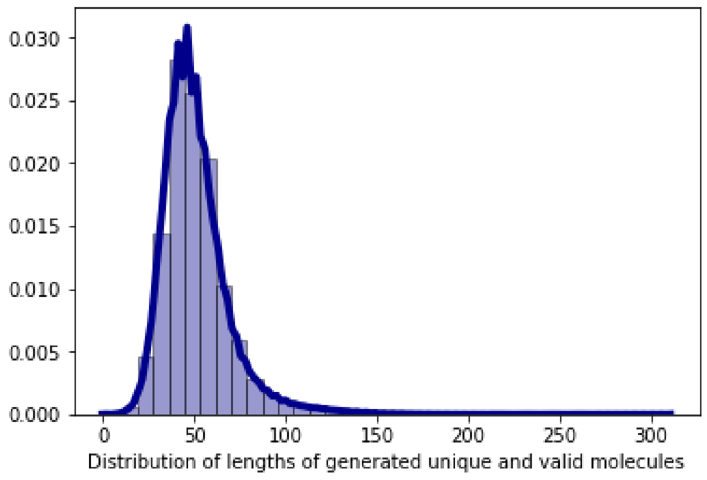


Figure 4.3: Distribution of lengths of generated molecules

## 4.2 Classified Molecules

Of the 2,245,495 molecules, 0.45% have been classified as non-toxic (compare Figure 4.4) which yields a set of 10,194 non-toxic molecules. An analysis of the distribution of non-toxic molecule lengths also shows that some of the outliers have been discarded as the maximum length decreased from 300 to around 200 (compare Figure 4.5).

The macro-average Area under the ROC curve (which can be seen in Figure 4.6 and 4.7) is 0.76. When compared to the leaderboard of the Tox21 challenge, the classifiers would have missed a leaderboard position in all categories by around 2% most of the time.

Examining the normalized distributions of assigned label probabilities (compare Figure 4.9) shows that for some classifiers (especially NR-AR-LBD and SR-ADAT5) the majority of toxic molecules have been misclassified as non-toxic. This is confirmed when looking at the False Negative Rates (FNR) for each assay type in Figure 4.8.

The mean False Negative Rate over all classifiers is 35.40 ± 15.03 which means that every fifth to second toxic molecule has been classified as non-toxic. The NR-AhR classifier has the lowest mean False Negative Rate of 14% which still means that roughly every seventh molecule is misclassified as non-toxic. The highest mean rate was produced by the NR-AR-LBD classifier which is around 65%, so nearly two thirds of toxic molecules of this type have been misclassified.

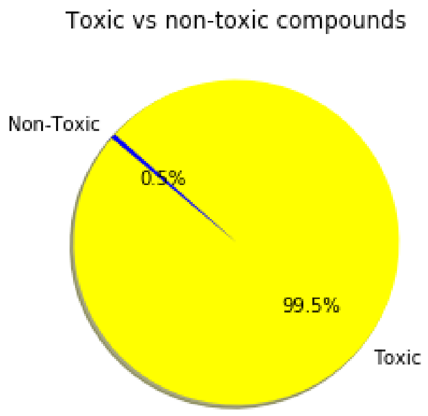


Figure 4.4: Ratio of toxic and non-toxic compounds in the library

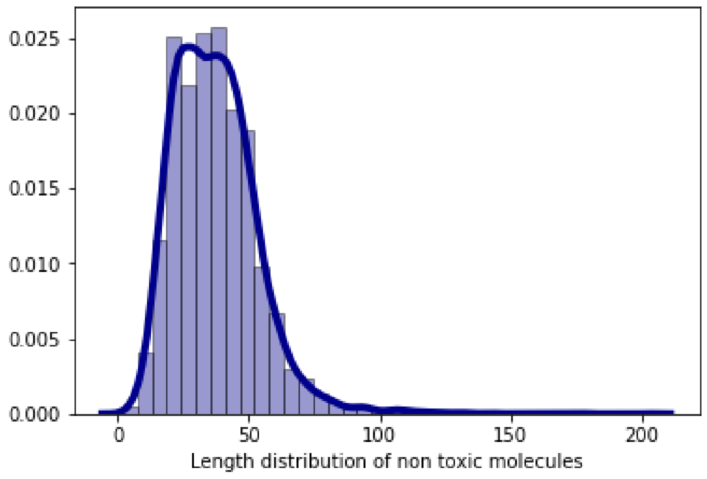


Figure 4.5: Distribution of lengths of non-toxic molecules

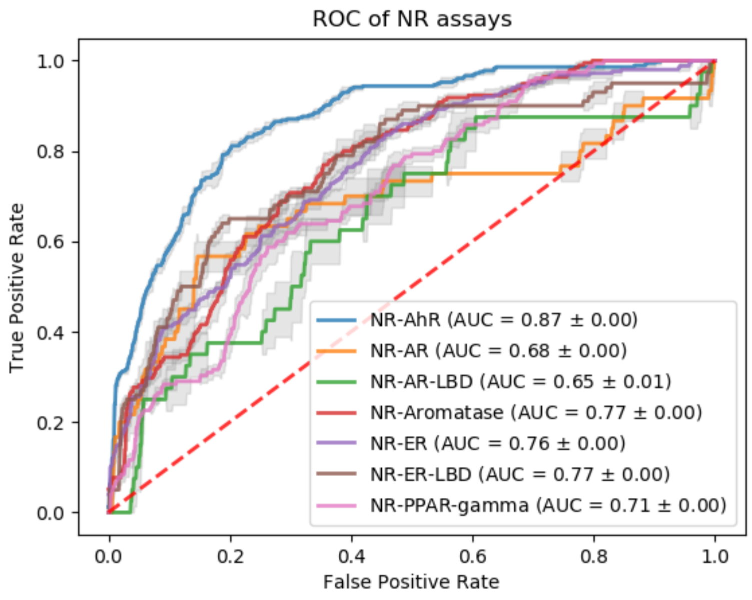


Figure 4.6: ROC of NR assays

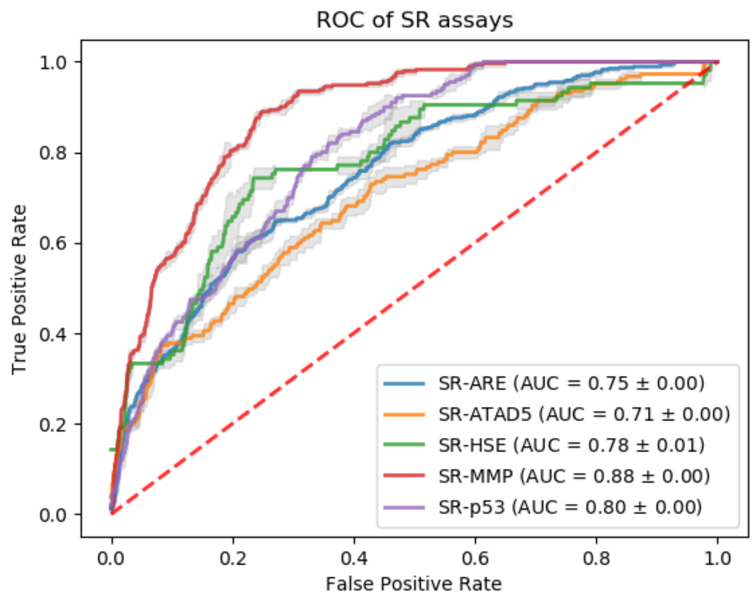


Figure 4.7: ROC of SR assays

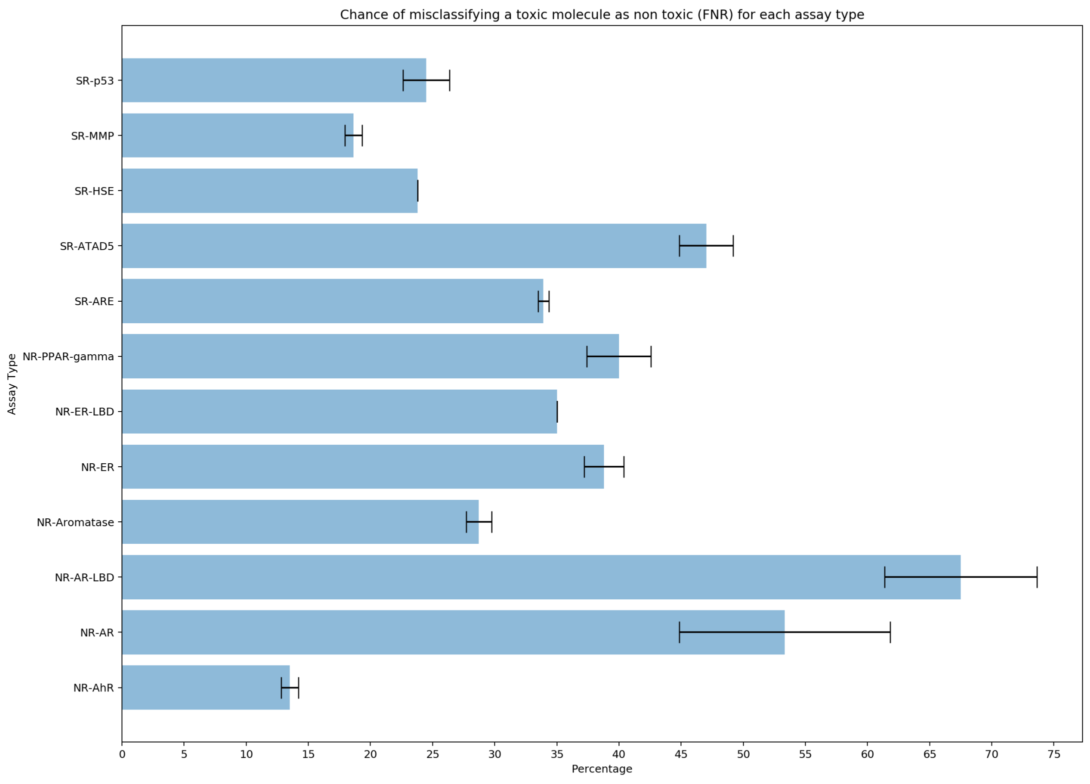
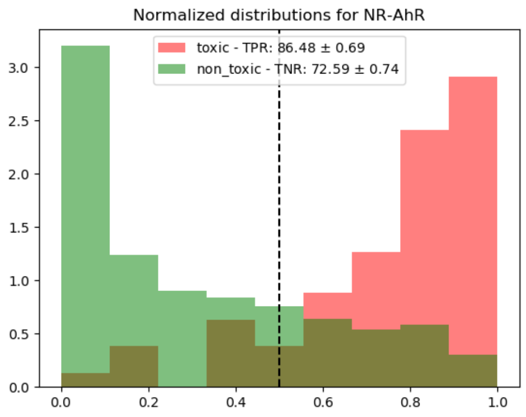
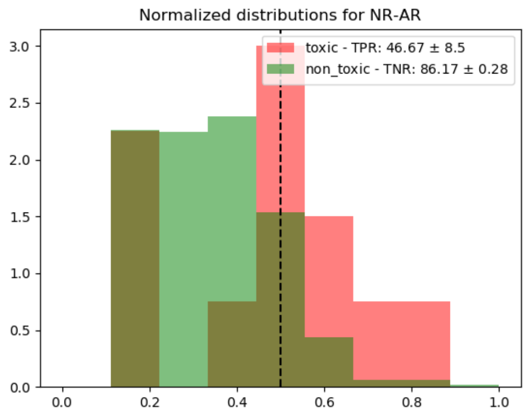


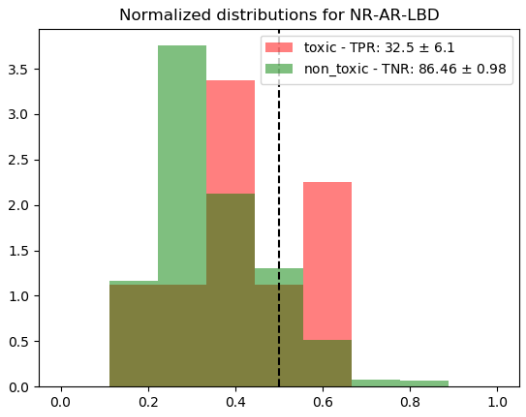
Figure 4.8: False Negative Rate for all assays



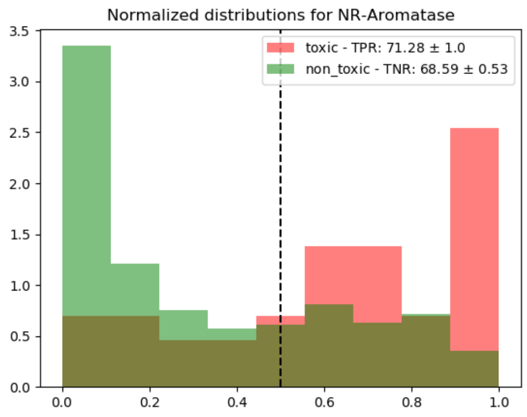
(a) NR-AhR



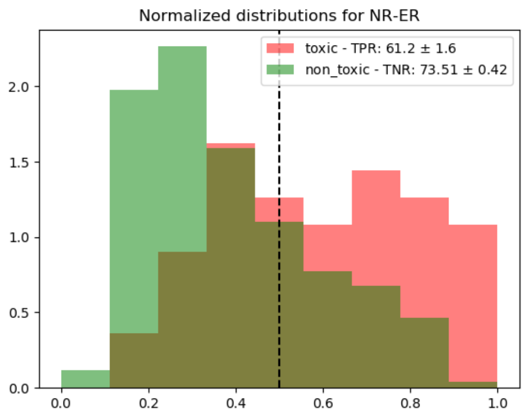
(b) NR-AR



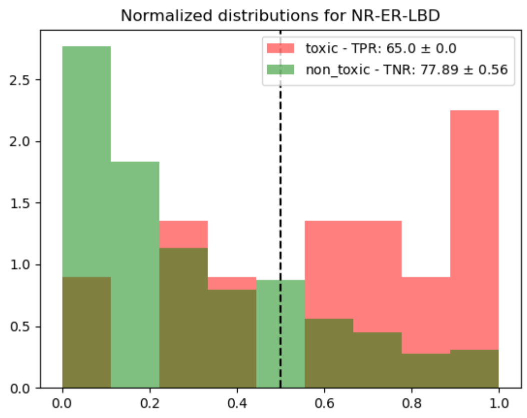
(c) NR-AR-LBD



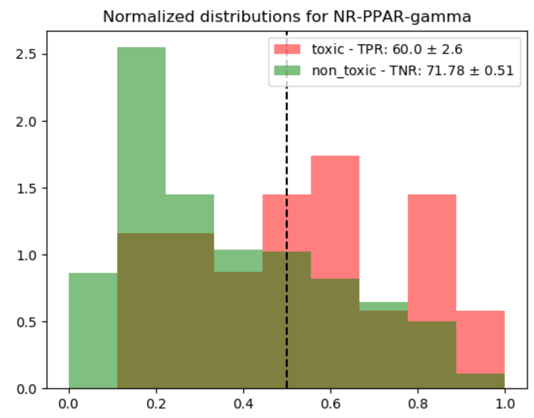
(d) NR-Aromatase



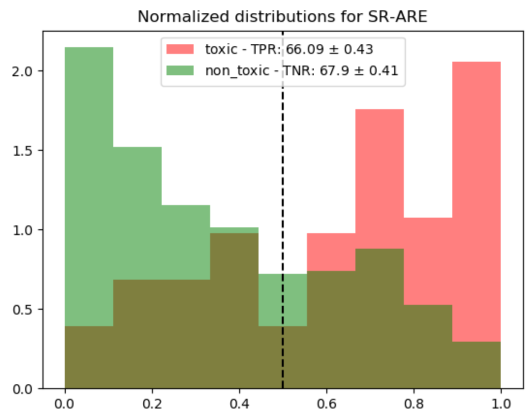
(e) NR-ER



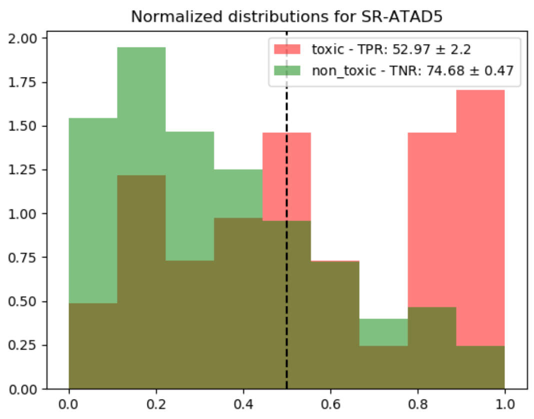
(f) NR-ER-LBD



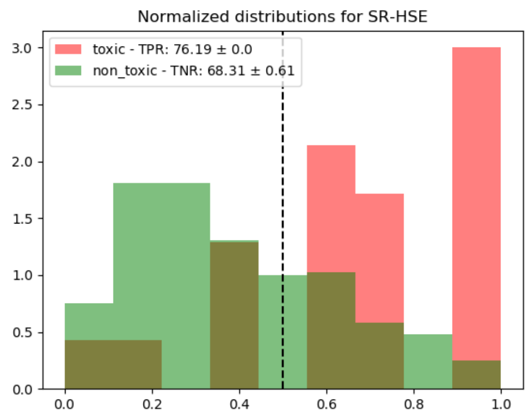
(g) NR-PPAR-gamma



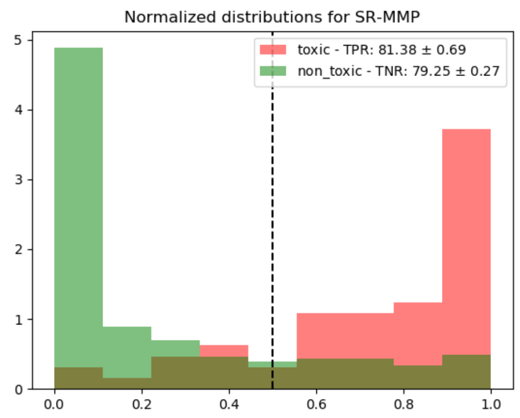
(h) SR-ARE



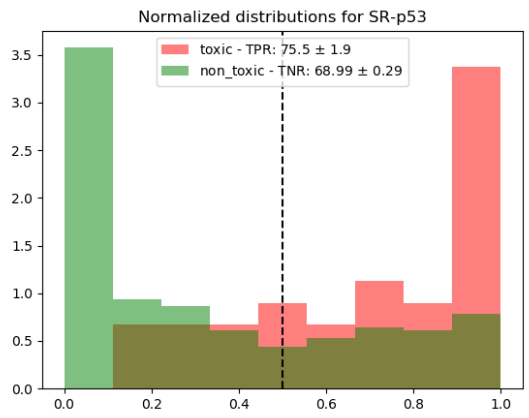
(i) SR-ADAT5



(j) SR-HSE



(k) SR-MMP



(l) SR-p53

Figure 4.9: Sensitivity (TPR) and Specificity (TNR) plots depicting the proportion of correctly and incorrectly classified molecules for each assay

# Chapter 5

# Discussion

The molecule generation process has been successful with 89.82% of generated molecules being valid and unique. The classification of said molecules needs further tuning especially when compared to other classifiers.

To improve the molecule generation process, a dictionary of all possible characters of the SMILES language could be provided. This would make the generator data set independent. For the training of the LSTM, second order gradient descent methods as well as regularization could improve the performance. An extensive hyperparameter grid search for both the generator and the classifier should also be performed to obtain the optimal hyperparameters.

As discussed in Segler et al.[4] transfer learning is an option if one wants to combat the generation of toxic molecules or produce focused molecule libraries. In that case the model trained in this thesis would serve as the base model and would be fine tuned on a smaller set of molecules which contain the required properties.

The performance of the molecule classifiers still has potential growth, which is evident when their performance is compared to the classifiers on the Tox21 leaderboard[19]. The classifiers trained in this experiment fell short of the top 10 for each assay by at least 1% and going up to 10% on the ROC AUC score.

Another issue is that the examination of the false negative rate showed that the misclassification for toxic molecules, depending on the assay type, ranges between 14% and 65%. This means the generated library most likely contains toxic molecules. One way of improving the molecule classifiers would be to collect more training samples, especially toxic ones. Another way would be to take the same approach as Mayr et al.[9] by enhancing the whole pipeline. Such measures would include adding further pre-processing, extracting more features from the available data, as well as introducing post-processing.

# Chapter 6

# Conclusion

The proposed molecule generation and classification pipeline has been implemented and has been successful in creating molecules. However there is still further room for improvement when it comes to the classification.
Due to the availability of millions of molecules via public libraries such as ChEMBL and standards such as SMILES, data driven methods have sufficient training data for molecule generation. Molecule classification seems feasible as well due to the Tox21 Program providing data and driving development.
LSTMs proved to be an excellent tool for the task of generating molecules. While a few minor improvements are possible for the molecule generator, the classifier still has much untapped potential. This becomes obvious when one compares its performance to the leaderboard entries of the Tox21 challenge. Further improvement is needed when it comes to its false negative rate, the rate of incorrectly classifying toxic molecules as non-toxic ones. This rate is considerably high with rates ranging between 14% and 66% depending on the assay. This results in the molecule library most likely containing toxic molecules.
To extend the work described here one could make the generator data set independent, by providing all possible letters of the SMILES language. It is also feasible to use the provided generator as a base model for transfer learning on a smaller set of compounds. This could yield a more focused library. To make the classifier more robust to misclassifying toxic molecules, one would either need more toxic training samples or a more sophisticated approach.
Out of the more than 2.5 million generated compounds, 2.25 million ($\sim$ 90% of the initial data set) remained after dropping invalid and duplicate molecules. Of those only 0.5% have been classified

as non-toxic, leaving 10,194 molecules in the library.

# List of Figures

# List of Tables

# Bibliography

[1] Melvin J Yu. “Druggable chemical space and enumerative combinatorics”. In: *Journal of Cheminformatics* 5.1 (Apr. 2013). DOI: 10.1186/1758-2946-5-19.

[2] Gisbert Schneider and Karl-Heinz Baringhaus. *Molecular design : concepts and applications*. Jan. 1, 2008.

[3] Marwin H. S. Segler and Mark P. Waller. “Modelling Chemical Reasoning to Predict and Invent Reactions”. In: *Chemistry - A European Journal* 23.25 (Jan. 2017), pp. 6118–6128. DOI: 10.1002/chem.201604556.

[4] Marwin H. S. Segler et al. “Generating Focused Molecule Libraries for Drug Discovery with Recurrent Neural Networks”. In: *ACS Central Science* 4.1 (Dec. 2017), pp. 120–131. DOI: 10.1021/acscentsci.7b00512.

[5] Anna Gaulton et al. “The ChEMBL database in 2017”. In: *Nucleic Acids Research* 45.D1 (Nov. 2016), pp. D945–D954. DOI: 10.1093/nar/gkw1074.

[6] Mark Davies et al. “ChEMBL web services: streamlining access to drug discovery data and utilities”. In: *Nucleic Acids Research* 43.W1 (Apr. 2015), W612–W620. DOI: 10.1093/nar/gkv352.

[7] National Center for Advancing Translational Sciences (NCATS). *Tox21 Operational Model*. Ed. by National Center for Advancing Translational Sciences (NCATS). Nov. 30, 2018. URL: https://ncats.nih.gov/tox21/about/operations.

[8] Ruili Huang and Menghang Xia. “Editorial: Tox21 Challenge to Build Predictive Models of Nuclear Receptor and Stress Response Pathways As Mediated by Exposure to Environmental Toxicants and Drugs”. In: *Frontiers in Environmental Science* 5 (Jan. 2017). DOI: 10.3389/fenvs.2017.00003.

[9] Andreas Mayr et al. "DeepTox: Toxicity Prediction using Deep Learning". In: *Frontiers in Environmental Science* 3 (Feb. 2016). DOI: `10.3389/fenvs.2015.00080`.

[10] Inc. Daylight Chemical Information Systems. *SMILES Simplified Molecular Input Line Entry System*. Ed. by Inc. Daylight Chemical Information Systems. June 21, 2019. URL: `https://www.daylight.com/smiles/`.

[11] Inc. Daylight Chemical Information Systems. *SMILES - A Simplified Chemical Language*. Ed. by Inc. Daylight Chemical Information Systems. June 21, 2019. URL: `https://www.daylight.com/dayhtml/doc/theory/theory.smiles.html`.

[12] Inc. Daylight Chemical Information Systems. *Fingerprints - Screening and Similarity*. Ed. by Inc. Daylight Chemical Information Systems. Sept. 6, 2019. URL: `https://www.daylight.com/dayhtml/doc/theory/theory.finger.html`.

[13] Greg Landrum. *RDKit: Open-source cheminformatics*. Ed. by Greg Landrum. Jan. 1, 2016. URL: `http://www.rdkit.org`.

[14] Sepp Hochreiter and Jürgen Schmidhuber. "Long Short-Term Memory". In: *Neural Comput.* 9.8 (Nov. 1997), pp. 1735–1780. ISSN: 0899-7667. DOI: `10.1162/neco.1997.9.8.1735`. URL: `http://dx.doi.org/10.1162/neco.1997.9.8.1735`.

[15] Python Software Foundation. *Python 3.0*. Ed. by Python Software Foundation. Sept. 8, 2019. URL: `https://www.python.org/`.

[16] Adam Paszke et al. "Automatic Differentiation in PyTorch". In: *NIPS Autodiff Workshop*. 2017.

[17] Eugine Kang. *Long Short-Term Memory (LSTM): Concept*. Ed. by Eugine Kang. Sept. 2, 2017. URL: `https://medium.com/@kangeugine/long-short-term-memory-lstm-concept-cb3283934359`.

[18] Tianqi Chen and Carlos Guestrin. "XGBoost: A Scalable Tree Boosting System". In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '16. San Francisco, California, USA: ACM, 2016, pp. 785–794. ISBN: 978-1-4503-4232-2. DOI: `10.1145/2939672.2939785`. URL: `http://doi.acm.org/10.1145/2939672.2939785`.

[19] National Center for Advancing Translational Sciences (NCATS). *Tox21 Leaderboards*. Ed. by National Center for Advancing Translational Sciences (NCATS). Jan. 12, 2015. URL: https://tripod.nih.gov/tox21/challenge/leaderboard.jsp.