A deep learning approach to multi-label mechanisms of action prediction for chemical compounds

Lars Bergh
STUDENT NUMBER: 2041774

THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE IN DATA SCIENCE & SOCIETY
DEPARTMENT OF COGNITIVE SCIENCE & ARTIFICIAL INTELLIGENCE
SCHOOL OF HUMANITIES AND DIGITAL SCIENCES
TILBURG UNIVERSITY

Thesis committee:

Dr. Giacomo Spigler
Dr. Dimitar Shterionov

Tilburg University
School of Humanities and Digital Sciences
Department of Cognitive Science & Artificial Intelligence
Tilburg, The Netherlands
January 2021

Contents

A deep learning approach to multi-label mechanisms of action prediction for chemical compounds

Lars Bergh

In drug discovery and drug repurposing, understanding Mechanisms of Action (MoA) is paramount. A MoA refers to the process by which chemical compounds bind to protein targets in the human body to interfere in a disease process. In prior research, MoA prediction was largely treated as a binary target problem. To understand side effects of drugs however, the use of multi-label MoA prediction is advantageous since many drugs can target multiple protein targets, some of which unwanted. In previous research, input data largely consisted of gene expression and protein sequence data while in vitro cellular viability data was largely excluded. To address these shortcomings, the following research question was formulated:

"What multi-label deep learning model and preprocessing pipeline is more effective in predicting Mechanisms of action for chemical compounds than a multi-layer perceptron with unprocessed cellular viability, genetic expression, dosing, and treatment duration data?"

To address the stated research question, a data set was acquired through a Kaggle coding competition. In this competition an organization called the "Broad Institute" (Institute 2020a) provided a data set through their "Connectivity Map" (Institute 2020b) project, which seeks to catalogue DNA to RNA transcription mediated by genetic and chemical disturbances. With this data set a deep learning framework was proposed that utilizes 2 multi-layer perceptrons (MLP) that combine their predictive power to obtain better MoA predictions than a single MLP. The first MLP is used to predict probabilities for each drug target for a given compound, while the second MLP predicts the number of drug targets that the drug has. Multiplying the probability matrix from the first MLP with a row weight from the second MLP results in the final prediction matrix.

Using a combination of a probability matrix with a row weight vector from 2 MLP improved the binary cross-entropy (BCE) loss from 0.01653 using the best prediction matrix to 0.01534 when compared to an ensemble model of 4 probability matrices and 3 row weight prediction vectors. The code used in the research is available at https://github.com/LarsBergh/MoA

1. Introduction

1.1 Goal of the research

The goal of this research was to propose an effective multi-label deep learning model and preprocessing pipeline for Mechanism of Action (MoA) prediction for cellular viability, gene expression, dosing and treatment duration data. A MoA refers to the ability of a drug-like compound to bind to a protein fold, which results in the interruption of a disease process. This interruption is caused by the binding event blocking the protein

fold from binding with other molecules. A MoA prediction therefore refers to a model prediction that is able to predict how a drug will target various protein folds. MoA prediction is relevant since it allows the effectiveness of drugs to be predicted without testing in an organism. The multi-labeled nature of MoA prediction methodologies enables scientists to understand the range of protein folds that a drug targets in the human body. This enables a deeper understanding of drug side effects since drug targets can be predicted for a given compound. Understanding the wide array of protein folds that a drug targets allows the treatment of complex diseases, since these treatments require multiple protein folds to be targeted at once. The prediction of MoA has been addressed extensively by the scientific community in the last few decades. State of the art MoA prediction methodologies have primarily used binary target prediction of a single drug target. In these models scientists try to find the drug that most predictably targets a specific protein fold.

1.2 Societal and scientific relevance

Drug discovery is an expensive and time consuming process that can cost over 1 billion dollars and 12 to 15 years to complete (Hughes et al. 2011). The drug discovery process typically requires various steps before a drug is brought to market. First, a disease is identified for which treatments are not sufficient or non-existent. Secondly, data is generated by academia or public companies to develop hypothesis. These hypothesis test for the inhibition or activation of a protein target that should in turn improve patient outcomes (Hughes et al. 2011). Thirdly, further validation is performed to provide scientific evidence in regards to the efficacy of targeting a certain protein fold to resolve a certain disease. If sufficient evidence is collected, an effort is launched to find a drug like substance that can reliably target the protein fold in an effective and safe manner (Hughes et al. 2011). Lastly, the selected drug-like substances proceed through pre-clinical and clinical development. If the drug is both efficacious and safe, it will be approved as a medicine (Hughes et al. 2011).

When drugs fail to get to market, drug safety or drug efficacy is usually lacking. This research attempts to addresses both problems, by providing a deep learning and data preprocessing framework that allows scientists to identify likely drug targets for a chemical compound. This aids both drug discovery and the understanding of drug side effects (Wen et al. 2017). Due to the fact that complex diseases require multiple drug targets to be targeted simultaneously, multiple drugs are often blended. Improved MoA prediction can therefore assist in efforts to prevent side effects since the entire set of protein targets of a drug blend that treats complex diseases could be predicted (Wen et al. 2017). These improvements could in turn result in faster drug discovery, leading to a higher probability of positive outcomes for patients with various diseases (Chen et al. 2018). Furthermore, drug side effects may be better understood since the wide range of protein targets of a drug can be predicted. These advances could alleviate disease induced suffering and costs related to the impact of disease on the human population.

1.3 Research questions

The goal of the research was to propose an effective multi-label deep learning and pre-processing MoA prediction framework for cellular viability, genetic expression, dosing, and treatment duration data. The following research question was addressed during the research:

"What multi-label deep learning model and preprocessing pipeline is more effective in predicting Mechanisms of action for chemical compounds than a multi-layer perceptron with unprocessed cellular viability, genetic expression, dosing, and treatment duration data?"

To answer the research question, a number of sub-questions were formulated. These are the following:

Sub-Rq 1: *Can model generalization be improved by dimesionality reduction on a baseline multi-layer perceptron when minimizing binary cross-entropy test loss?*

Since the data set contains 875 columns of data to predict MoA, generalization to unseen data may be compromised by the large amount of potentially irrelevant features. Dimensionality reduction methodologies may remove these irrelevant features and improve performance since the feature quality for the deep learning model is higher. This in turn could aid in the reduction of BCE test loss in the final model.

Sub-Rq 2: *What feature scaling regiment is appropriate to maximize multi-label mechanism of action model performance by minimizing binary cross-entropy test loss?*

For a Neural Network (NN) to best predict classes in a classification problem, it requires classes to be separable based on feature values in the training data. To provide the best separability between classes, feature scaling regiments can be applied to transform each feature value based on a given function. Choosing a proper scaling regiment can therefore improve class separability which in turn improves the NN performance by decreasing BCE test loss in the final model.

Sub-Rq 3: *What multi-layer perceptron structure can better predict multi-label mechanism of actions to reduce binary cross-entropy loss compared to a single multi-layer perceptron?*

When the input data is properly formatted, a single multi-layer percepron (MLP) can be utilized to perform multi-label MoA prediction. It is however not obvious that this architecture is optimal. Therefore, a simple MLP baseline model was compared to an architecture that utilizes multiple MLPs. Choosing a proper MLP structure may lower BCE test loss of the final model by improving the prediction matrix output.

Sub-Rq 4: *Can hyperparameter tuning be used to reduce binary cross-entropy loss compared to a baseline multi-layer perceptron for multi-label mechanism of action prediction?*

Since it is unlikely that MoA predictions can be represented by a convex function, various hyperparameter settings could be tweaked to improve the likelihood of converging towards a lower local minimum when optimizing for BCE test loss. Utilizing hyperparameter tuning methodologies may therefore improve the odds of a model locating a lower local minimum and therefore decrease BCE test loss in the final model.

Sub-Rq 5: *Can ensemble models be used to decrease binary cross-entropy loss compared to a single model for multi-label mechanism of action prediction?*

When a deep learning algorithm outputs a prediction matrix given optimized hyperparameters, it may be the case that a model ensemble can improve performance by averaging multiple prediction matrices. This ensemble model may therefore reduce BCE test loss which would improve the MoA predictions in the final model.

By answering the research questions that were previously stated, a novel methodology could be developed to predict multi-label drug targets on cellular viability and gene

expression data with deep learning. The resulting algorithm could aid in drug discovery and drug repurposing efforts by providing a novel MoA prediction avenue.

1.4 Findings

Based on the conducted experiments, it can be concluded that Principal Component Analysis (PCA) and upsampling increases BCE test loss. These methods were therefore not used in the final model. To scale cellular viability and gene expression columns, the Scikit-learn Quantile Transformer with a uniform output distribution was found to perform best on the baseline model. This transformer was therefore used in the final model. The best performing MLP architecture combined 2 MLP ensembles in which a prediction matrix was multiplied with a row weight vector. To tune hyperparameters, 50 parameter sets were tested for each MLP and an ensemble of 4 prediction matrices and 3 row weight vectors was found to minimize BCE test loss best. More detailed information on the test results can be found in the "Results" section.

2. Related Work

2.1 Research context

The acquisition of novel treatments for diseases is a pivotal goal in biomedical research (Emig et al. 2013; Bleakley and Yamanishi 2009). Less than 400 of the estimated 30.000 genes in the human genome, encode for proteins that are used as drug targets (Emig et al. 2013). For a protein to qualify as a drug target, it requires folds in its structure where a drug-like molecule can attach to a binding site. It is useful for a drug to bind to such a binding site if the protein structure plays a role in a disease process (Bull and Doig 2015). When a drug is able to target a protein fold, it is referred to as a Mechanism of Action (MoA).

Drug discovery usually requires a vast amount of biological tests to be conducted. These so called "assays" measure toxicity, inhibition and activation of proteins (Mayr et al. 2018; Sturm et al. 2020). Past drug discovery has primarily followed a one molecule one target scheme in which scientists search for the most specific drug to act on one drug target for a single disease (Chen et al. 2016). With this approach, the potential of a certain protein to function as a drug target is first tested in vitro and later in entire organisms. Even though this system has been effective for some simple diseases, more complex diseases require more than 1 drug target to be targeted simultaneously for effective treatment (Chen et al. 2016). Utilizing models to predict multiple MoA may therefore prove useful in selecting drugs for more complex diseases. Furthermore, understanding side effects of drugs may be improved due to the ability of the model to predict both wanted and unwanted drug targets.

2.2 Similar research

Understanding how chemical compounds effect diseases is vital in drug discovery and drug repurposing. The use of deep learning to predict MoA is a relatively new phenomenon. Various data driven techniques to predict MoA have been used in prior decades, but papers regarding the use of deep learning methodologies have have largely appeared after 2014. Numerous papers in recent years have attempted to predict MoA

with various input data and modeling techniques. Table 1 provides a brief overview of the input data and modeling approaches of widely cited papers in this area.

Table 1
Prominent papers that predict MoA.

Paper	Model input & prediction type	Model types
(Xie et al. 2017)	Drugs & Gene expression, Binary	MLP, SVM, CNN
(Zheng et al. 2018)	Drugs & Gene expression, Binary	LSTM, CNN
(Gao et al. 2018)	Drugs & Gene expression, Binary	LSTM, CNN
(Feng et al. 2018)	Drugs & Proteins, Binary	CNN
(Lee, Keum, and Nam 2019)	Drugs & Proteins, Multi-label	CNN
(You, McLeod, and Hu 2019)	Drugs & Proteins, Binary	MLP

MoA prediction in the literature is largely represented as a binary classification problem as can be observed in table 1. Model inputs usually consist of consist of raw protein sequences, drug chemistries and drug induced gene expression as features. The targets that are predicted are binary MoA targets that are known for the drugs in the data set.

In a study, (Xie et al. 2017) attempted to use multi-label Support Vector Machine (SVM), Convolutional Neural Networks (CNN) and a MLP with Softmax outputs to predict MoA. The SVM validation accuracy in the study reached 20%, while a CNN approach and a Softmax MLP approach reached 40% and 70% validation accuracy respectively. Predicting MoA with NNs has often resulted in non satisfactory prediction accuracy. To address this, (Zheng et al. 2018) proposed a combination between CNN and a LSTM (long short term memory) network with a dense output layer. These so called DTI-RCNN networks demonstrated precision scores over 92% compared to a baseline MLP that achieved a precision of 84% on drug chemistry and gene expression data (Zheng et al. 2018). The use of CNN and LSTM models was further explored by (Gao et al. 2018) who encoded drug chemistry with CNN and embedded protein target features through a LSTM embedding. The LSTM embedding was performed on amio acid sub-sequences after which both inputs were fed through an MLP that predicted the MoA. The model outperformed comparable models with accuracy levels exceeding 85% on unseen protein data without feature engineering or domain knowledge. The prediction of MoA is further complicated by the so called "cold-target" problem, which refers to the prediction of target proteins that do not appear in the training set (Feng et al. 2018). To solve this problem, the so called "Protein And Drug Molecule interaction prEdiction framework" (PADME) was developed, which successfully takes protein and compound information to predict MoA (Feng et al. 2018). In one instance where multi-label target prediction was performed, CNN were trained on proteins to predict binding cites (Lee, Keum, and Nam 2019). The CNN were trained on amino acid sub sequences and successfully located binding sites for MoA to take place (Lee, Keum, and Nam 2019). In the area of drug repurposing (You, McLeod, and Hu 2019) effectively used l1 regularization and a MLP to predict MoA with a 75% accuracy. To achieve this, the model used drugs and tripeptides as input features. Tripeptides represent proteins in subsections of 3 amino acids.

Besides the prediction of MoA, deep learning has been used in pharmacology for drug repurposing by predicting the drug rather than the drug target. In a study

performed by (Mei and Zhang 2019) a multi-label framework was proposed to find new drugs for known targets and repurposing old drugs for new uses. The supervised learning model used known genetic and drug target information to predict other drug targets that an existing drug may have. The model correctly predicted at least 1 type of drug for 84.9% of target genes using l2-regularization and logistic regression with a chance level of 0.16% (Mei and Zhang 2019).

All studies mentioned in table 1 agree on the effectiveness of gene expression or protein sequences as input features to predict MoA. Furthermore, there is agreement on the ineffectiveness of simple Machine Learning models to predict MoA due to the complex non-linear relationships that represent the relationship between drugs and their targets (Xie et al. 2017; Zheng et al. 2018). To reduce the complexity of feature spaces of protein sequences and drug chemistries to learn MoA, CNN are a well supported methodology (Feng et al. 2018; Lee, Keum, and Nam 2019; Gao et al. 2018).

2.3 Literature shortcomings & research implications

The literature regarding the prediction of MoA differs from this research in both input data and modeling techniques. During this research, cellular viability and gene expression were used to explore a new avenue of MoA by including cellular viability data besides gene expression data. This differs from previous research that mainly used protein sequences, drug chemistry, gene expression or a combination of these as input features. In the literature the problem of MoA prediction is mostly treated as a binary classification problem to predict whether a drug binds to a given drug target. In this research however, the problem is treated as a multi-label classification problem which allows a wide array of targets to be predicted for a single drug. Complex diseases require multiple drug targets to be targeted simultaneously, which requires pharmaceutical companies to use a blend of active compounds that each target different protein folds. The more compounds that are used together, the higher the chances of side effects due to off-target protein interactions (Takarabe et al. 2012; Whitebread et al. 2005; Blagg 2006). Improving MoA prediction could enable better predictability of all protein targets that a drug may have and therefore help understand side effects of drugs. This is especially true for complex diseases since multiple compounds may be used for one disease. Proper MoA predictions may therefore improve drug discovery efforts since side effects are a significant contributor to drugs failing to come to market.

By answering the research questions stated in the introduction, a novel methodology could be developed to predict multi-label drug targets which in turn could aid in the discovery of new drugs for more complex disease states due to its multi-labelled nature. Furthermore, the cellular viability and gene expression data provide an avenue for new chemicals to be tested in petri dishes after which drug target predictions could be performed based on these results. For this research a data set published by harvard and MIT was be used. This data set was published to Kaggle and provides a combination of cellular viability and messenger RNA (mRNA) expression data for thousands of chemical compounds. More information regarding the origin and contents of the data can be found in the "Experimental setup" section.

3. Methods

To answer the previously stated research questions, a set of experiments was proposed as displayed in figure 1.

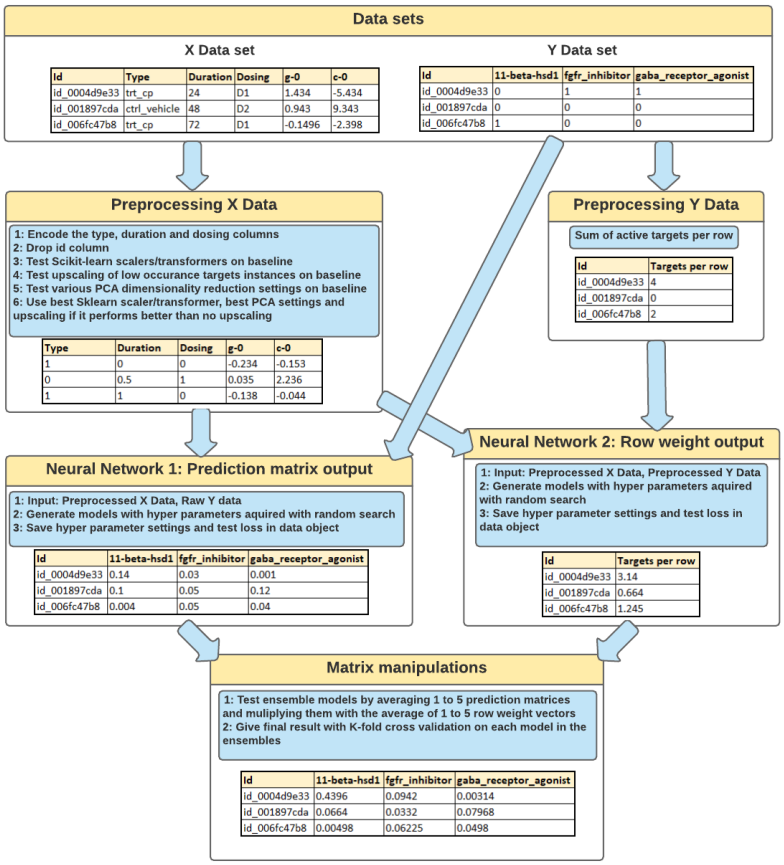

Figure 1
MoA prediction experiments pipeline.

3.1 Dimensionality reduction

To test dimensionality reduction performance on the X data set, the Scikit-learn PCA class was utilized to compress the cellular viability and gene expression columns into a lower dimensional space. For both groups of columns, various amounts of PCA components were returned and then tested on a baseline MLP to compare BCE test loss.

3.2 Input data scaling

To acquire a scaling regiment for cellular viability and gene expression columns, 6 types of Scikit-learn scalars and transformers were tested. Each scaling regiment was applied on the X data set and the performance of the various scaling regiments was tested on

a baseline MLP. To address class imbalances in the representation of each drug target, upsampling of low representation targets was tested and compared to a baseline MLP without upsampling.

3.3 Multi-layer perceptron architecture

To test what deep learning architecture is most effective in minimizing BCE loss, a combination of experimentation and first principals thinking was applied. First principals thinking would let one conclude that there are 2 components required for a perfect multi-label prediction matrix. The first component would consist of a model that outputs maximum probability to the targets that have a value of 1.0 in the Y data set. This however only works if there is a total of 1 target for a given row since there is only 1.0 probability to distribute with a Softmax probability output layer in the MLP. The second component therefore needs to account for the actual amount of targets in a given row so that the total probability in that row matches the number of targets with a value of 1.0. This would require a row weight per predicted row to account for the amount of targets in a row.

To address the research question, a deep learning model that can output an N by M prediction matrix was proposed. This model utilizes 2 different MLP that combine their prediction into a single prediction matrix of N by M, where N represents the amount of drug administration rows and M represents the amount of target columns. The first MLP receives a training data matrix with P rows of drug administrations and Q feature columns. The input layer size is equivalent to the amount of feature columns Q in the data set. Through a series of hidden layers and computations, the model outputs probabilities for each target class represented in the output columns M in the Y data set with a Softmax output layer. The Softmax function then transforms values into probabilities utilizing equation 1.

$$\sigma(z)_i = \frac{e^{\mathbf{z}_i}}{\sum_{j=1}^{K} e^{\mathbf{z}_j}} \tag{1}$$

For each target prediction vector $\mathbf{z}$ that the MLP predicts for a single compound, the Softmax function is applied to each component of this vector to obtain probabilities for each drug target. For each single prediction in the prediction vector $\mathbf{z}$, the natural number e is squared by the current component i in the vector $\mathbf{z}$, representing the numerator. In the denominator, the natural number e is squared by all components in vector $\mathbf{z}$ independently, where the current component to be calculated is denoted by j until j reaches the last index K. The sum of these squared components results in the final denominator value. The result of one Softmax calculation represents a single probability that a single drug target is active for a given compound. The Softmax function is applied to each component in the N by M prediction matrix, transforming the numerical values into probabilities.

The second MLP attempts to adjust the prediction row probabilities in the N by M prediction matrix with a weight. This allows cumulative row probability to be adjusted from a total of 1.0 to the actual probabilities. If there are 3 active targets in a given row, the second MLP would ideally predict 3 as the row weight. This weight could then

be used to multiply the row probabilities by 3. To obtain the row weights, the second MLP is trained on the sum of active targets per row of the Y data set. The MLP tries to optimize its performance by minimizing the Mean Absolute Error (MAE) between the predicted and actual amount of active targets per row. The MAE is calculated utilizing equation 2.

$$MAE = \frac{\sum_{i=1}^{n} \mid y_i - x_i \mid}{n} \tag{2}$$

For each row P in the X training data set, a single output row weight is predicted. The MAE is calculated by taking the sum of the absolute difference between each prediction y_i and true value x_i. The summed difference of each computation is then divided by the amount of rows denoted by n. This computation is performed by iterating over all n components, where the current index is represented by i.

3.4 Hyperparameter tuning

During the training process of both MLP, random parameter sets were sampled from a predefined feature space. These selected hyperparameter sets were then used to generate deep learning models of which the model performance was recorded. For each hyperparameter set, the hyperparameters and the obtained BCE loss was stored in a data object that was then sorted based on BCE loss.

3.5 Ensemble models

To further optimize model results, various row weight vector ensembles and prediction matrix ensemble models were tested. For each of the ensembles, a given number of best performing hyperparameter sets were pulled from the hyperparameter data object and used to train MLP. To obtain performance improvements over a single model, 25 ensemble models were tested in which 1 through 5 prediction matrices were averaged and then multiplied with 1 through 5 row weight vectors that were also averaged by performing element-wise mean operations. Each combination of average prediction matrices and average row weight vectors were tested, resulting in 25 experiments of which the BCE loss was logged. For all tests the MLP were computed with K-fold cross validation to get more accurate model performance metrics.

4. Experimental Setup

In this section, the data set, preprocessing, modeling and hyperparameter tuning is discussed. To find the utilized programming language, libraries and their respective versions that were used during the research see Appendix 2.

4.1 Data set source

The data set was sources from a Kaggle coding competition created by Massachusetts Institute of Technology (MIT), Harvard Laboratory for Innovation Science (LISH) and the NIH Common Funds Library of Integrated Network-Based Cellular Signatures

(LINCS) (for Innovation Science at Harvard 2020b). These three organizations are part of the "Broad Institute" (Institute 2020a) which has provided the data set through their "Connectivity Map" (Institute 2020b) project, which seeks to catalogue DNA to RNA transcription mediated by genetic and chemical perturbations (Institute 2020b).

The data set contains cellular viability and messenger RNA (mRNA) transcription abundance data for various chemical perturbations as measured by the L1000 essay (John Davis 2016). mRNA can be understood as the link between the sequences of human DNA and the proteins that can be synthesized based on this genetic material. mRNA acts as a copy of a piece of relevant genetic material, created in the nucleus of the cell through a process called transcription. After transcription the mRNA travels from the cell nucleus to the ribosomes of the cell which act as protein synthesis factories. The L1000 essay measured the mRNA abundance within cells for 978 genes by introducing chemical perturbations in cells and measuring the effect on the mRNA that was produced by these cells as a function of these disturbances (John Davis 2016). Previous attempts at multi-label MoA prediction relied mainly on only gene expression data relied rather than cellular viability and gene expression data. This data set is unique in the sense that it includes both types of data.

4.2 Data set description

The provided data set contains 2 csv files that were used during the research. The "train features" and "train targets scored" csv files both consist of 23814 rows of data. The "train features" data set contains 875 columns which are the columns used for training, while "train targets scored" contains 205 columns of binary output targets. The number of columns in each data set excludes the "sig id" column that functions as an unique identifier for each row. Each of the rows represent one sample of drug administration to living cells. Table 2 displays the data types and descriptions of the "train features" data set.

Table 2
Data types and descriptions of the "train features" data set.

Columns name	Description of values	Data type
Sig id	Unique identifyer per sample	String
Cp type	Ctl vehicle = control group / trt cp = treatment group	String
Cp time	Treatment duration (24, 48, 72 hour exposure)	Int
Cp dose	Drug dosing (high - D1, low - D2)	String
G (g-0 to g-771)	Gene expression values (772 columns)	Float
C (c-0 to c-99)	Cellular viability per cell type (100 columns)	Float

To provide a clearer picture of the "train features" data, table 3 displays 5 rows of the data set in its unprocessed form. Because the data set has 876 columns, the gene expression and cellular viability columns have been reduced to 1 of each.

Table 3
Example rows of the "train features" data set.

Sig_id	Cp_type	Cp_time	Cp_dose	G-0	C-0
Id_000644bb2	trt_cp	24	D1	1.0620	-0.0600
Id_000779bfc	trt_cp	72	D1	0.0743	0.0927
Id_000a6266a	trt_cp	48	D1	0.6280	-0.1312
Id_fffb70c0c	trt_cp	24	D2	-1.3260	0.2144
Id_fffc1c3f4	ctl_vehicle	48	D2	0.3942	1.0650

The "train targets scored" data set contains 23814 rows of data matching the rows in the "train features" data set. For each drug administration row, 205 columns have either a value of 1 if the drug binds to the target, or a 0 if it does not. Examples of these drug targets include the 5-alpha reductase inhibitor, 11-beta-hsd1 inhibitor, acat inhibitor, acetylcholine receptor agonist and the acetylcholine receptor antagonist. Table 4 describes the types of targets in the "train targets scored" data set.

Table 4
Drug target groups in the "train targets scored" data set.

Target group	Amount of target columns	Description
Inhibitor	112	Decreases enzyme activity
Antagonist	32	Minimizes receptor response
Agonist	28	Activates receptors
Activator	5	Increases enzyme activity
Other	28	Other classifications

Table 4 demonstrates that 177 out of the 205 drug targets are either enzyme or receptor drug targets. Enzymes speed up biological processes like digestion. The so called Inhibitor targets are enzymes that can have their activity decreased due to a drug binding to it, while activators acts the exact opposite by kick-starting or speeding up the use of the enzyme. Receptors differ from enzymes because they control the flow of signals that flow in and out of the cell like a gate-keeper in the cellular wall. The antagonist class decreases the receptor its ability to transmit specific signals in or out the cell, while the agonist stimulates the receptor to produce an effect in the cell. The "other" category contains drug targets that cannot be classified in one of the other 4 categories but have properties making them act antiviral, antibiotic, or anticonvulsant for example.

4.2.1 Data distributions. Prior to modeling and data processing, a random state of 0 was set to ensure reproducible results of all experiments. To best predict drug targets for a given set of compounds, information regarding target and data distributions can inform how transforming the input of the model can improve performance. These insights can be divided into distributions of both the targets and the input data. In figure 2 the data distributions of both cellular viability and gene expression columns were plotted for 4 randomly selected columns in both categories.

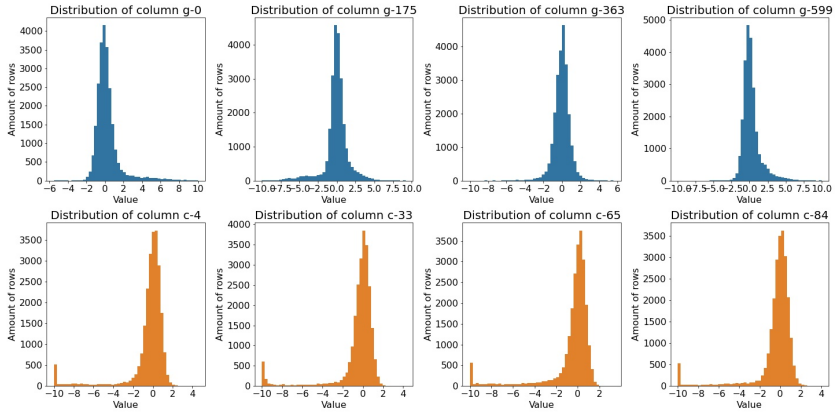

Figure 2
Value distributions of cellular viability and gene expression columns.

The distributions in figure 2 seem to suggest relatively normal distributions with some skew and some leptokurtic behaviour. To measure the normality of the cellular viability and gene expression columns, skewness and kurtosis distributions were created. To provide further insight, the distribution values were binned into categories in separate bar charts to demonstrate skewess and kurtosis levels based on categorically defined measures. Table 5 describes the cutoff points used for the categorical measurements of skewness and kurtosis as specified by (Gravetter and Wallnau 2014).

Table 5
Categorical cutoff points for kurtosis and skewness bins.

Skew category	Kurtosis category	Value range
Skew left	Platykurtic	$\leq$ -2
Symmetric	Mesokurtic	-2 $<$ X $<$ 2
Skew right	Leptokurtic	$\geq$ 2

based on the bin ranges described in table 5, the gene expression and cellular viability column skewness and kurtosis were plotted in figure 3.

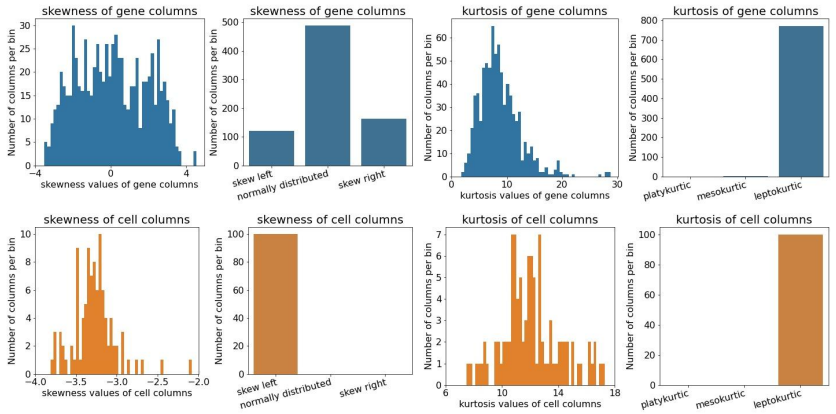

Figure 3
Skewness and Kurtosis distributions of cellular viability and gene expression columns.

Based on figure 3 it can be concluded that both gene expression and cellular viability columns are highly leptokurtic, meaning that the distribution is relatively tall compared to its width and therefore lacks normality. In regards to skewness there seems to be a discrepancy in how cellular viability and gene expression columns behave. The gene expression columns seem mostly normally distributed, while cellular viability columns are all skewed left. With these insights, scaling schemes were tested forcing these columns towards various distributions, and therefore manipulate their utility as input columns for the deep learning models.

To provide insight into drug target distributions, both row and column derived drug target distributions were plotted. Figure 4 displays how often each number of drug targets occurs across the rows of drug administration in the entire data set.

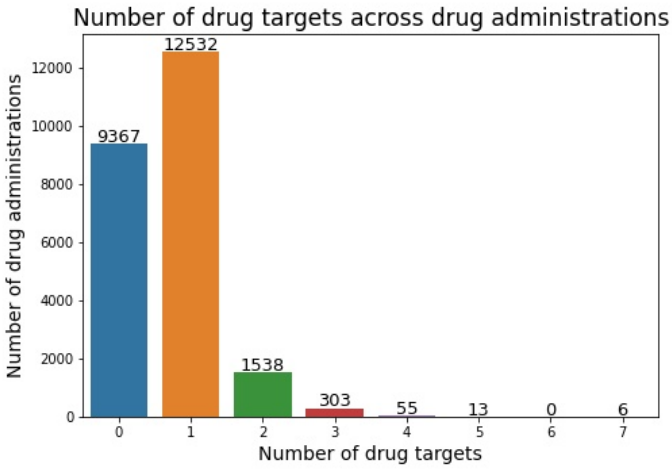

Figure 4
Number of active drug targets for drug administrations rows.

From figure 4 it can be derived that 39.3%, 52.6% and 8.0% of rows, in the data set require 0.0, 1.0 or more than 1.0 probability to be distributed across its rows since each active drug target should have a value of 1.0. Since a default MLP with a Softmax output will only attribute 1.0 probability across a row of targets, almost half of predictions will overshoot or undershoot this 1.0 value. This is due to the fact that the total amount of probability that is being distributed across a given row is 1.0 if no row weights are used. Solutions to this problem are discussed in the modeling section.

To display frequencies of the various drug targets, a column wise summation of all active targets in the Y data set was plotted in figure 5. This figure displays the top 50 most frequently occurring drug targets in the Y data set.

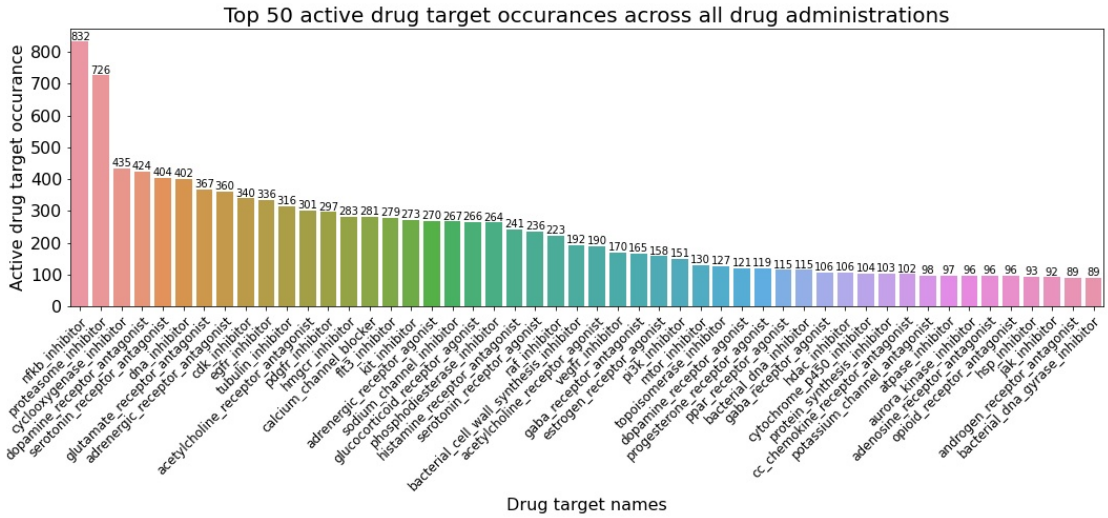

Figure 5
Cumulative active drug targets across all drugs in the data set.

The degree to which each of the targets is represented in the data set is highly unbalanced. This is demonstrated by the fact that the top 50 target account for 68.5% of all active targets in the data set, while the bottom 50 drug targets accounts for 3.3% of active drug targets. Attempts at dealing with these class imbalances can be found in the "Preprocessing" section.

4.2.2 Preprocessing. Prior to modeling, several preprossessing steps were taken to improve the model input. First, the data was loaded and the id columns of both X and Y data were dropped. Since the data did not contain any missing data, no missing data imputation strategies were required. The cp_type, cp_time and cp_dose columns were then encoded with a mapping. In this mapping scheme each of mentioned column values were replaced with values between 0 and 1 as described in figure 6.

Table 6
Mapping utilized for cp_type, cp_time and cp_dose columns.

Column	Mapping
Cp_type	ctl_vehicle: 0, trt_cp: 1
Cp_time	24: 0, 48: 0.5, 72: 1
Cp_dose	D1: 0, D2: 1

All other columns that relate to cellular viability and gene expression were manipulated with Scikit-learn scalars and transformers. To acquire an effective data manipulation scheme, 6 Scikit-learn scalars and transformers from the Scikit-learn preprocessing class were tested on a baseline MLP. Table 7 describes these scalars and transformers:

Table 7
Utilized Scikit-learn scalars and transformers.

Scalar name	Description
Standardscalar	Scales values to a zero mean with unit variance
MinMaxscalar	Scales values between 0 and 1
Robustscalar	Remove median, scale to interquartile range
QuantileTransformer normal	Scales to normal distribution, robust to outliers
QuantileTransformer uniform	Scales to uniform distribution, robust to outliers
PowerTransformer	Scales values to a more normal distribution

Due to the high dimensionality of the input data, dimensionality reduction measures were tested on a baseline MLP. To test how well the model would perform with various degrees of dimensionality reduction, PCA was used by implementing the PCA function of Scikit-learn. Since cellular viability and gene expression columns contain different types of information, PCA was performed on both groups of columns separately. In total 6 different PCA settings were tested where each test contained progressively more cellular viability and gene expression columns. The results of these experiments can be found in the "Results" section.

To split the data, the train test split function from Scikit-learn was used to split the data into a train set of 80% and a test set of 20%. During the modeling, the K-fold cross validation function from Scikit-learn was used to split the data into train and validation folds after which models were generated for each k-fold. The test set was then used at the end to evaluate each model with unseen data. During all experiments only 2 folds were used due to limits in the available computational power.

4.3 Modeling

To measure the performance of various preprocessing and advanced modeling strategies against some baseline, a simple 2 layer MLP was used as a baseline with a Softmax output. This baseline model was created with a Keras implementation of Tensorflow MLPs. Table 8 displays the Keras model parameters used to create the baseline MLP.

Table 8
baseline MLP hyperparameters.

Parameter	Value
Number of hidden layers	2
Hidden layer neurons	64
Hidden layer activation	Relu
Output layer neurons	206
Output layer activation	Softmax
Optimizer	Adam
Loss function	Binary Crossentropy
Model fit batch size	4
Epochs	15
Patience with early stopping	1 epoch

4.3.1 Hyperparameter selection & tuning. The hyperparameters tuning for the two MLP was performed through a 3 step process for each model. First, the feature space was defined. This required lists of parameters to be created from which random parameter sets could be randomly sampled by a function. Each of the parameter ranges was limited based on early experimentation.

The number of layers and the number of neurons were limited to a maximum of 5 and 192 respectively due to early experimentation with hyperparameter settings and the tendency towards overfitting with more model complexity. MLP without dropout were also tested and resulted in overfitting, which is why the parameter was included in the feature space. When selecting hidden layer activation options, a subset of mostly non-linear activation functions was chosen based on early experimentation success with non-linear activation functions. To define possible optimizers, a list of widely used optimizers was created. Table 9 displays the feature space that was used for the parameter selection process.

Table 9
Features space used for random sampling of hyperparameter.

Parameter	Values
Number of hidden layers	2, 3, 4, 5
Hidden layer neuron count	64, 96, 128, 160, 192
Dropout	0.15, 0.20, 0.25, 0.3, 0.35, 0.4
Hidden layer activation	elu, relu, sigmoid, softplus, softsign
Optimizer	nadam, adam, SGD, adamW, rmsprop

For the SGD optimizer a learning rate of 0.05 and a momentum of 0.98 were chosen. For the AdamW optimizer a weight decay of 1e-4 was used. After defining the feature space, a function was used to randomly select unique parameter sets from the feature space and run MLP with these hyperparameters. 50 of these models were generated and saved into an object that stored the parameters and the test loss of the model. This object was then sorted based on the test loss. Each of these models was created with only a 2-fold cross validation due to a constraint in compute.

Finally, ensemble models were created. To test various ensemble models, 1 through 5 of the best performing models were used in ensembles for each of the MLP. Since there were 2 MLP, a total of 25 different ensembles were tested. To understand the degree to which BCE loss could theoretically be improved by the 2 MLP, a small sensitivity analysis was performed. In this analysis the predicted row weights were compared to the actual row weights to establish the impact a perfect row weights has on BCE test loss. This result allows one to infer that all other BCE loss is created by the prediction matrix, rather than the row weights. The results of these experiments is discussed in the "Results" section.

4.4 Evaluation metrics

The performance of the created algorithms is judged by evaluating a prediction matrix. For each sample in the data set the probability that each of the 205 MoA targets had a positive response was predicted. Since the Y test set contains 20% of the entire data set of 23814 rows, a 4763 row by 205 column prediction matrix of probabilities between 0 and 1 was created by the model. Each of the values in this matrix represent the probabilities of

an active drug target for a given chemical compound. The performance of the solution is based on the average value of the logarithmic log function applied to each row of predicted values compared to the actual values, as stated by the formula below (for Innovation Science at Harvard 2020a). This formula is usually referred to as binary cross-entropy loss or log loss and is defined by equation 3.

$$BCE = -\frac{1}{M}\sum_{m=1}^{M}\frac{1}{N}\sum_{i=1}^{N}[y_{i,m}\ log(\hat{y}_{i,m}) + (1-y_{i,m})\ log(1-\hat{y}_{i,m})] \tag{3}$$

To compute the the BCE loss for a single target in row i and column m, a calculation is made that adds up two values for that given row and column. The first value multiplies the actual value $y_{i,m}$ with the base e logarithm of the predicted value $\hat{y}_{i,m}$. The second value subtracts the actual value $y_{i,m}$ from 1 and multiplies this result with the base e logarithm of 1 minus the predicted value $\hat{y}_{i,m}$. To perform these calculations for a total BCE loss, the calculations are repeated and summed for each row i in N total rows and for each m in M total columns. Multiplying the acquired total value with -1 gives the final BCE loss.

5. Results

To encode the dosing, drug administration duration and treatment type columns, a simple mapping was used to scale values between 0 and 1. This mapping was used on the X data set for all further experiments.

5.1 Dimensionality reduction

To test dimensionality reduction effects on model performance, various PCA settings were tested and the BCE loss was measured. Table 10 displays the BCE losses for each tested combination of cellular viability and gene expression components returned by the PCA class. The amount of components refers to the amount of columns that were left after compressing the columns of both cellular viability and gene expression. Each of the settings was tested on the baseline MLP.

Table 10
baseline MLP BCE test loss given varying degrees of PCA dimensionality reduction.

Number of cell columns	Number of gene columns	BCE Test loss
5	25	0.01784
10	50	0.01778
15	75	0.01772
35	150	0.01783
50	200	0.01789
100	772	0.01770

Based on table 10 it can be concluded that the inclusion of all cellular viability and gene expression columns leads to the lowest BCE test loss. The differences between the various component selections are small however. In all further experiments that were

performed, all cellular viability and gene expression columns were kept.

5.2 Input data scaling

To attain the best scaling scheme for cellular viability and gene expression columns, 6 Scikit-learn scalars and transformers were applied to the X data and tested on the baseline MLP. For each test the BCE test loss was logged as displayed in table 11.

Table 11
Baseline BCE test loss given various Scikit-learn scalars and transformers.

Scikit-learn scalar/transformer name	Binary cross-entropy loss
QuantileTransformer uniform	0.01727
Standardscalar	0.01763
PowerTransformer	0.01766
Robustscalar	0.01775
QuantileTransformer normal	0.01788
No scalar	0.01803
MinMaxscalar	0.01934

During the experiment, the Quantile Transformer resulted in the lowest BCE loss on the test set. Therefore it was used in all further experiments and the final model.

To address the unbalanced drug target representation across all rows in the data set, all drug targets that were represented in less than 100 rows, were up-sampled to 100 rows. Without upsampling, the BCE test loss was 0.01818 while with upsampling the the BCE test loss was 0.02364 on the baseline MLP. Due to the decrease in performance, upsampling was not used in the final model.

5.3 Multi-layer perceptron architecture

An architecture consisting of 2 MLP outperformed a single MLP. The first MLP predicts probabilities for each of the targets, while the second MLP predicts the number of targets for a given compound, which acts like a weight for the row. Multiplying the probability matrix from the first MLP with the row weight vector of the second MLP results in the final prediction matrix. The BCE loss of the best hyperparameter optimized prediction matrix was 0.01653 while the best prediction matrix multiplied with the best row weight vector resulted in a BCE loss of 0.01560.

5.4 Hyperparameter tuning

To select hyperparameters for the MLP, 50 randomly sampled model hyperparameter sets for each MLP were selected. The resulting BCE test loss and the hyperparameter sets were saved in an object and sorted from lowest to highest BCE test loss. Table 12 displays the 5 best performing hyperparameter sets for the probability prediction MLP. The AdamW optimizer in the table used a weight decay of 1e-4.

Table 12
Best hyperparameter sets for probability matrix MLP.

Models	Layers	Activation	Neurons	Dropout	Optimizer	BCE test loss
1	2	Elu	96	0.2	AdamW	0.01653
2	2	Elu	128	0.25	AdamW	0.01658
3	2	Sigmoid	96	0.25	AdamW	0.01662
4	4	Elu	64	0.2	Adam	0.01662
5	3	Softplus	128	0.15	nAdam	0.01663

To find the best row weight MLP hyperparameter sets, a similar approach was used. 50 hyperparameter sets were randomly sampled from the same feature space and used to train the MLP. The resulting MAE losses and the hyperparameter sets were then saved into a data object that was then sorted based on MAE test loss. Table 9.1 displays the 5 best performing hyperparameter sets for the row weight prediction MLP.

Table 13
Best hyperparameter sets for row weight MLP.

Model	Layers	Activation	Neurons	Dropout	Optimizer	MAE test loss
1	2	Sigmoid	192	0.3	nAdam	0.3954
2	2	Softsign	64	0.15	Adam	0.3993
3	2	Sigmoid	96	0.15	AdamW	0.3998
4	2	Sigmoid	96	0.2	Adam	0.4015
5	2	Sigmoid	128	0.25	AdamW	0.4017

5.5 Ensemble models

With these model results, ensembles of both the probability matrix MLP and the row weight MLP were tested up to 5 models per ensemble. This resulted in a total of 25 tests because all combinations between 1 and 5 models were tested for both the ensemble probability matrix and row weight predictions. The acquired BCE test losses found during the 25 experiments ranged between 0.01560 using only 1 model in both ensembles and 0.01534 when averaging 4 probability matrices and 3 row weight predictions. Without the use of the row weight vectors, the model ensemble performance ranged from a BCE loss of 0.01653 with 1 prediction matrix to 0.01606 with an ensemble of 4 models. Appendix 1 displays the full list of results of the ensemble results. To understand the maximum improvements that could be realized with each MLP, a sensitivity analysis was performed on the BCE test loss contributions to the final result. By comparing actual row weights from the Y test data to the predicted row weights, the BCE loss could be reduced from 0.01534 to 0.01362, which yields an 11.2% improvement compared to the best ensemble model. This implies that the remaining 0.01362 BCE loss can all be accounted for by the predictions made by the MLP that outputs probability matrices. This therefore represents 88.8% of BCE loss.

6. Discussion

The goal of the research was to propose a deep learning model and preprocessing pipeline for multi-label MoA predictions for chemical compounds. The framework provides a methodology to use cellular viability and gene expression data, treatment duration, treatment group and dosing as inputs to predict whether a drug bind to a set of drug target.

Cellular viability and gene expression columns were scaled with the Scikit-learn Quantile Transformer since it had the lowest BCE test error on the baseline MLP. Upsampling low occurrence targets and using dimensionality reduction through PCA were tested, but both decreased performance and were therefore not used in the final model. With the preprocessed data 2 MLP were trained to create a combined prediction matrix. The first MLP predicts the probability for each drug target using a Softmax output layer, while the second MLP predicts a row weight that is multiplied with each Softmax output vector. A combination of the 4 best prediction matrices and the 3 best row weight vectors was found to be optimal for the final ensemble.

It was surprising to find that a second MLP improved prediction matrix performance by predicting a row weight, since it was not clear that the X data set provided the needed information to predict row weights. This finding implies that the data gives some indication of a chemical its ability to bind to drug targets overall.

The research and its results were limited in a number of ways. First, k-fold cross validation could only be applied in 2 folds due to limitations in compute, potentially leading to a less accurate representation of actual performance in comparison to a model with more folds. Second, only PCA was used for dimensionality reduction while the use of a variantial autoencoders might provide a better representation of the data and therefore may perform better on BCE test loss. Third, only random search was used to tune hyperparameters. Using more advanced hyperparameter optimization techniques like Bayesian hyperparameter tuning may improve results. Bayesian hyperparameter tuning uses a probability model to select hyperparameter sets which may lead to improved BCE test loss. Fourth, the data set was limited in the sense that the addition of more features may help improve row weight and prediction matrix predictions. Perfectly optimizing row weight predictions could result in a 11.2% improvement in BCE test loss while improving MoA probability predictions could result in no more than of 88.8% reduction in BCE test loss. Some of the 88.8% of BCE test loss may be reduced accounting for the unbalanced drug classes. This may be achieved by weighing the losses that can be attributed to each training batch by the relative occurrence of targets in the training batch.

The results are not yet applicable to accurately predict multiple drug targets. In the literature, accuracy levels are mostly calculated based on binary accuracy for a given target (Mei and Zhang 2019; Xie et al. 2017), rather than the accuracy of an entire row of probabilities for a set of targets. The difficulty of defining an accuracy level based on these probabilities is due to the cutoff points must be defines for a target so it can be set to the value 1. A possible solution may be to use the rounded row weight prediction vectors as a proxy for the number of targets that should be active in a given row. If a row weight of 0.65 for example is rounded to 1, it could imply that the highest probability value should be rounded to 1, with all other values set to 0. Since this creates a binary matrix, accuracy and other metrics like recall and precision can be calculated. The difficulty with a metric like accuracy in the current implementation, is that 99.65%

of the prediction matrix should be 0 and only 0.35% should be a 1. This makes poorly performing models look good on an accuracy bases, due to the large frequency of zeros predicted by the model.

The proposed model can be used as a blueprint for the prediction of multi-label drug target prediction given cellular viability, genetic expression, drug dosing, treatment group and treatment duration data. Before this model can be used properly in the field however, improvements must be made to ensure that some comparable forms of accuracy can be computed relative to other models.

7. Conclusion

To provide a deep learning framework to predict MoA the following research question was answered:

"What multi-label deep learning model and preprocessing pipeline is more effective in predicting Mechanisms of action for chemical compounds than a multi-layer perceptron with unprocessed cellular viability, genetic expression, dosing, and treatment duration data?"

To answer the main research question, the following sub questions were answered:

Sub-Rq 1: *Can model generalization be improved by dimesionality reduction on a baseline multi-layer perceptron when minimizing binary cross-entropy test loss?*

Since the X data set of 875 columns has high dimensionality, the Scikit-learn PCA class was used to test BCE loss on various amounts of reduced dimensionality. This however, degraded performance slightly in all tested combinations of cell viability and gene expression columns. Using all cellular viability and gene expression columns led to a BCE loss of 0.01770, while the best performing dimensionality reduction consisted of 15 cell viability columns and 75 gene expression columns with a BCE loss of 0.01772 on the baseline MLP. These results demonstrate that reductions in data dimensionality are negligible to slightly worse when attempting to improve BCE loss in MoA predictions through dimensionality reduction. Model generalization therefore could not be improved in the tested PCA dimensionality reduction scenarios.

Sub-Rq 2: *What feature scaling regiment is appropriate to maximize multi-label mechanism of action model performance by minimizing binary cross-entropy test loss?*

To find a well performing scaling scheme for the cellular viability and gene expression columns, 6 Scikit-learn scaling and transformation schemes were tested. The Quantile Transformer with a uniform output distribution performed best by reducing BCE loss from 0.01818 without the transformation to 0.01755 with the transformation to on the baseline MLP. From these experiments it can be concluded that using the Quantile Transformer is an appropriate scaling scheme for cellular viability and gene expression columns when minimizing BCE loss. To address class imbalances in the representation of each drug target, upsampling of low representation targets was tested. Upsampling the low frequency targets to 100 occurrences degraded performance from a BCE loss of 0.01818 before upsampling to 0.02364 after upsampling. It can therefore be concluded that the use of upsampling decreases performance due to an increase in BCE test loss. The use of upsampling is therefore not an appropriate method to minimize BCE loss when predicting MoA.

Sub-Rq 3: *What multi-layer perceptron structure can better predict multi-label mechanism of actions to reduce binary cross-entropy loss compared to a single multi-layer perceptron?*

A combination of 2 MLP was proposed in which the first MLP predicts probabilities for each of the targets for a given compound. The second MLP then predicts the number of active targets that a compound has. The number of targets predicted by the second MLP acts like a weight for the row probabilities given by the first MLP. Multiplying the probability matrix from the first MLP with the row weight vector of the second MLP gives the final prediction matrix.

When comparing the best prediction matrix of a single MLP with the best prediction matrix multiplied by the best row weights, a BCE loss of 0.01653 and 0.01560 are achieved respectively. This means that the MLP architecture that includes the row weight predictions outperforms the prediction matrix without the row weights. It can therefore be concluded that the architecture with the 2 MLP it preferred over the single MLP due to its decreased BCE loss.

Sub-Rq 4: *Can hyperparameter tuning be used to reduce binary cross-entropy loss compared to a baseline multi-layer perceptron for multi-label mechanism of action prediction?*

To tune the hyperparameters of both MLP, 50 randomly selected model parameter sets for each MLP were sampled and used for model training. The best performing model on the probability prediction MLP produced a BCE loss of 0.01653, compared to 0.01818 BCE loss for the baseline model. From these improvements it can be concluded that hyperparameter tuning through the use of random search is appropriate when improving model performance over the baseline model by reducing BCE test loss.

Sub-Rq 5: *Can ensemble models be used to decrease binary cross-entropy loss compared to a single model for multi-label mechanism of action prediction?*

To improve upon the probability matrix output MLP, ensembles of both the probability matrix MLP and the row weight MLP were tested up to 5 models per ensemble. Before the creation of the MLP, the Quantile Transformer was applied while upsampling and PCA were left out due to performance degradation on the baseline MLP. The best performing model utilizing a single probability prediction MLP produced a BCE loss of 0.01653, while an ensemble of 4 models performed best with a BCE loss of 0.01606. When applying the row weight vectors to the best prediction matrix ensemble, the best performance was achieved with an ensemble of the 3 best row weight vectors resulting in a BCE test loss of 0.01534. It can therefore be concluded that the use of ensemble models provides improved performance over the use of single models when minimizing BCE test loss.

These previously stated conclusions form the entire data processing and modeling pipeline required to more effectively predict MoA than a single MLP without data processing. This conclusion can only be drawn with the use of cellular viability, gene expression, treatment duration, dosing and treatment group data as model input.

Using a stack of 2 MLP to predict probabilities and row weight for multi-label drug targets could aid in drug target prediction by adjusting probabilities to match the total probability expected in a given row. This could allow cutoff points to be set above which a target could be considered active with a value of 1. This would not work if the row probabilities are not adjusted for the number of targets in a given row. This approach allows target probabilities to be interpreted as a 1.0, if a certain probability threshold is achieved. Achieving these cutoff points would not be possible if row probabilities were not scaled based on the amount of active drug targets in a row. These techniques could provide scientists with better models to predict multiple drug targets for compounds, increasing chances that the full range of effects of a compound can be understood.

To expand on the deep learning framework that was proposed, various improvement can be made. First, more accurate losses can be calculated by running the models with more than the 2 folds that were used due to compute constraints. Second, more advanced hyperparameter tuning could potentially have increased performance of the model due to smart hyperparameter selection techniques. An example of such techniques is Bayesian hyperparameter tuning that puts emphasis on exploring hyperparameter spaces that seem promising based on its internal probability model of the objective function.

A significant challenge in minimizing BCE loss is correctly predicting the number of targets a given drug has in that data set. The addition of extra features to optimize the row weight prediction MLP could help improve overall prediction matrix performance by making the row weights more accurate. This in turn would improve the model usability since it would allow for probability cut-off points at which drug targets could be considered active.

Finally, class balancing methods could be used by scaling the loss associated with each instance in its training batch to its relative occurrence in that batch. This ensures that less frequently occurring targets still have a decent impact on the back propagation process, preventing overfitting on frequently occurring classes. This in turn may reduce BCE test loss by improving the probability matrix prediction MLP.

8. Acknowledgements

I would like to thank the the Connectivity Map project participants for providing the data to Kaggle so I could perform my research. Without this multi-label prediction data set I could not have obtained the results that I have. Furthermore, I would like to thank my supervisor Dr. G. Spigler for always answering my questions and providing helpful feedback and suggestions. Lastly, I would like to thank all researchers in the field of MoA prediction on which my research was built.

References

Blagg, Julian. 2006. Structure–activity relationships for in vitro and in vivo toxicity. *Annual Reports in medicinal chemistry*, 41:353–368.

Bleakley, Kevin and Yoshihiro Yamanishi. 2009. Supervised prediction of drug–target interactions using bipartite local models. *Bioinformatics*, 25(18):2397–2403.

Bull, Simon C and Andrew J Doig. 2015. Properties of protein drug target classes. *PloS one*, 10(3):e0117955.

Chen, Ruolan, Xiangrong Liu, Shuting Jin, Jiawei Lin, and Juan Liu. 2018. Machine learning for drug-target interaction prediction. *Molecules*, 23(9):2208.

Chen, Xing, Chenggang Clarence Yan, Xiaotian Zhang, Xu Zhang, Feng Dai, Jian Yin, and Yongdong Zhang. 2016. Drug–target interaction prediction: databases, web servers and computational models. *Briefings in bioinformatics*, 17(4):696–712.

Cournapeau, David. 2020. scikit-learn machine learning in python.

Emig, Dorothea, Alexander Ivliev, Olga Pustovalova, Lee Lancashire, Svetlana Bureeva, Yuri Nikolsky, and Marina Bessarabova. 2013. Drug target prediction and repositioning using an integrated network-based approach. *PLoS One*, 8(4):e60618.

Feng, Qingyuan, Evgenia Dueva, Artem Cherkasov, and Martin Ester. 2018. Padme: A deep learning-based framework for drug-target interaction prediction. *arXiv preprint arXiv:1807.09741*.

Gao, Kyle Yingkai, Achille Fokoue, Heng Luo, Arun Iyengar, Sanjoy Dey, and Ping Zhang. 2018. Interpretable drug target prediction using deep neural representation. In *IJCAI*, volume 2018, pages 3371–3377.

Google-Brain. 2020a. Tensorflow.

Google-Brain. 2020b. Tensorflow addons.

Gravetter, F.J. and L.B. Wallnau. 2014. *Essentials of Statistics for the Behavioral Sciences*. Wadsworth, Cengage Learning.

for Innovation Science at Harvard, Laboratory. 2020a. Mechanism of action (moa) prediction - evaluation.

for Innovation Science at Harvard, Laboratory. 2020b. Mechanism of action (moa) prediction - overview.

Hughes, James P, Stephen Rees, S Barrett Kalindjian, and Karen L Philpott. 2011. Principles of early drug discovery. *British journal of pharmacology*, 162(6):1239–1249.

Hunter, John D. 2020. Matplotlib: Python plotting.

Institute, Broad. 2020a. Broad institute.

Institute, Broad. 2020b. Unravel biology with the world's largest perturbation-driven gene expression database.

John Davis, David E. Peck, Willis Read-Button. 2016. What is the l1000 assay?

Lee, Ingoo, Jongsoo Keum, and Hojung Nam. 2019. Deepconv-dti: Prediction of drug-target interactions via deep learning with convolution on protein sequences. *PLoS computational biology*, 15(6):e1007129.

Mayr, Andreas, Günter Klambauer, Thomas Unterthiner, Marvin Steijaert, Jörg K Wegner, Hugo Ceulemans, Djork-Arné Clevert, and Sepp Hochreiter. 2018. Large-scale comparison of machine learning methods for drug target prediction on chembl. *Chemical science*, 9(24):5441–5451.

McKinney, Wes. 2020. Pandas - python data analysis library.

Mei, Suyu and Kun Zhang. 2019. A multi-label learning framework for drug repurposing. *Pharmaceutics*, 11(9):466.

Oliphant, Travis. 2020. Numpy.

van Rossum, Guido. 2020. Welcome to python.org.

Sturm, Noé, Andreas Mayr, Thanh Le Van, Vladimir Chupakhin, Hugo Ceulemans, Joerg Wegner, Jose-Felipe Golib-Dzib, Nina Jeliazkova, Yves Vandriessche, Stanislav Böhm, et al. 2020. Industry-scale application and evaluation of deep learning for drug target prediction. *Journal of Cheminformatics*, 12:1–13.

Takarabe, Masataka, Masaaki Kotera, Yosuke Nishimura, Susumu Goto, and Yoshihiro Yamanishi. 2012. Drug target prediction using adverse event report systems: a pharmacogenomic approach. *Bioinformatics*, 28(18):i611–i618.

Waskom, Michael. 2020. Seaborn: statistical data visualization.

Wen, Ming, Zhimin Zhang, Shaoyu Niu, Haozhi Sha, Ruihan Yang, Yonghuan Yun, and Hongmei Lu. 2017. Deep-learning-based drug–target interaction prediction. *Journal of*

Proteome Research, 16(4):1401–1409. PMID: 28264154.

Whitebread, Steven, Jacques Hamon, Dejan Bojanic, and Laszlo Urban. 2005. Keynote review: in vitro safety pharmacology profiling: an essential tool for successful drug development. *Drug discovery today*, 10(21):1421–1433.

Xie, Lingwei, Song He, Yuqi Wen, Xiaochen Bo, and Zhongnan Zhang. 2017. Discovery of novel therapeutic properties of drugs from transcriptional responses based on multi-label classification. *Scientific reports*, 7(1):1–11.

You, Jiaying, Robert D McLeod, and Pingzhao Hu. 2019. Predicting drug-target interaction network using deep learning model. *Computational Biology and Chemistry*, 80:90–101.

Zheng, Xiaoping, Song He, Xinyu Song, Zhongnan Zhang, and Xiaochen Bo. 2018. Dti-rcnn: New efficient hybrid neural network model to predict drug–target interactions. In *International Conference on Artificial Neural Networks*, pages 104–114, Springer.

List of Figures

List of Tables

9. Appendices

9.1 Appendix I: Binary cross-entropy for model ensembles

Matrices	Row weights	BCE with modeled row weight	BCE with perfect row weight
1	1	0.015595	0.013855
1	2	0.015558	0.013855
1	3	0.015563	0.013855
1	4	0.015558	0.013855
1	5	0.015576	0.013855
2	1	0.015486	0.013738
2	2	0.015450	0.013738
2	3	0.015454	0.013738
2	4	0.015449	0.013738
2	5	0.015455	0.013738
3	1	0.015375	0.013618
3	2	0.015339	0.013618
3	3	0.015343	0.013618
3	4	0.015338	0.013618
3	5	0.015342	0.013618
4	1	0.015390	0.013639
4	2	0.015354	0.013639
4	3	0.015357	0.013639
4	4	0.015353	0.013639
4	5	0.015357	0.013639
5	1	0.015391	0.013630
5	2	0.015355	0.013630
5	3	0.015359	0.013630
5	4	0.015354	0.013630
5	5	0.015358	0.013630

9.2 Appendix II: Programming language, packages and software versions

Language/package	Type of use	Version	Source
Python	Programming language	3.8.4	(van Rossum 2020)
Numpy	Data manipulation	1.18.5	(Oliphant 2020)
Pandas	Data manipulation	1.1.3	(McKinney 2020)
Seaborn	Visualisation	0.11.0	(Waskom 2020)
Matplotlib	Visualisation	3.3.2	(Hunter 2020)
Tensorflow	Deep Learning	2.3.1	(Google-Brain 2020a)
Sklearn	PCA, Cross validation	0.23.2	(Cournapeau 2020)
Tensorflow addons tuner	AdamW optimzer	0.11.2	(Google-Brain 2020b)