Stat254 Final Project: Predicting hERG Blocking Molecules

Kaitlin Smith
Department of Statistics
University of California, Berkeley
Berkeley, CA 94720
`kaitlin_smith@berkeley.edu`

1 Introduction

The coordination of the heart's beating relies on human ether-a-go-go genes (hERG). However, some molecular compounds block this vital gene, leading to severe side effects when these molecules are present in drugs. The ability to predict hERG liability in the early stages of drug development is crucial to reduce the risk of cardiotoxicity related affects in later drug development stages.

This project will explore three graph neural network (GNN) architectures for predicting whether or not a particular molecule is hERG blocking. Similar structures for the architectures will be implemented, while the graph network layers themselves are changed, in hopes of exploring whether or not various GNN layers are better suited to this problem than others. Although high predictive performance is desired from these models, it is not enough. Controlling the false negative rate is a key component to make this application reliable in a real-world setting.

To address this, conformal risk control will be implemented to guarantee that the probability of a molecule being classified as non-hERG blocking, when it actually is, will be at most 10%. Combining both the predictive power of neural networks with the probabilistic guarantees of conformal risk control creates a framework suited for the healthcare industry.

2 Data

The first consideration when developing models for molecule prediction is handling how to present the molecules both in human-interpretable format and in machine interpretable format. Typically, molecules are depicted in diagrams, which provide an effective format to depict the atoms in the molecule, the type of bonds, and the structure between the atoms. However, this visual format is not conducive to data storage or modelling. SMILES (simplified molecular-input line-entry system) strings were developed in the 1980s to tackle the data storage problem of molecules. SMILES strings encode molecular structure into short ASCII strings, and are a common data storage type for molecules.

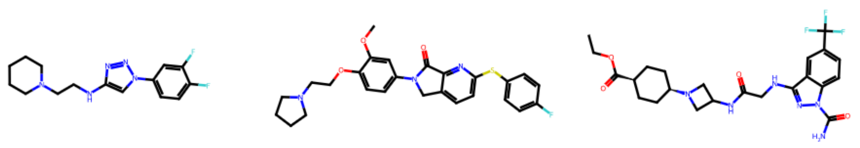


Figure 1: Examples of molecules from TDC dataset.

The dataset of molecules used in this project was obtained from Therapeutics Data Commons [3], and includes the SMILES strings of 13,445 molecules. The dataset was constructed by combining data from five different data sources including: the BindingDB database (3056 hERG blockers, 3039 non-blockers), ChEMBL bioactivity database (4859 hERG blockers, 4751 non-blockers), and others derived from literature, including 8,749 hERG blockers and 5,552 non-blockers [4]. These molecules are not conducive to modelling in their original format, as alpha-numeric strings. Various methods are needed to convert the strings into feature vectors the models can utilize.

2.1 SMILES Processing

Each SMILEs string is broken down into various components using the RDkit package. First, each SMILEs string is checked for validity, although no invalid strings were found in this dataset. Next, the atoms are extracted from the string, which represent the nodes in the graph representation. The atom type, the number of heavy neighbors, the formal charge of the atom, the hybridisation type, whether or not the atom is a ring, if it is aromatic, the atomic mass, and the atomic radius are also extracted. These atomic features are then one-hot-encoded and concatenated together to create an atomic-level feature vector for each node within the graph. Next, the graph structure is found by obtaining the adjacency matrix for the molecule.

2.2 Exploratory Data Analysis

In addition, it is important to examine summary statistics for the various molecular and atomic features that the model will consider.

Table 1: Summary Statistics for Molecular Level Features

Feature	Mean ± Std (Non hERG)	Mean ± Std (hERG)
NumAtoms	30.70 ± 6.37	31.08 ± 5.94
NumBonds	67.60 ± 14.40	68.69 ± 13.53

Table 2: Summary Statistics Atomic Level Features

Feature	Mean ± Std (Non hERG)	Mean ± Std (hERG)
NumHeavyNeighbors	2.20 ± 0.72	2.21 ± 0.69
FormalCharge	0.00 ± 0.05	0.00 ± 0.05
AtomicMassScaled	0.02 ± 0.03	0.02 ± 0.03
VDWRadiusScaled	0.28 ± 0.10	0.29 ± 0.10
CovalentRadiusScaled	0.06 ± 0.06	0.06 ± 0.06
NumHydrogens	0.82 ± 0.89	0.85 ± 0.89

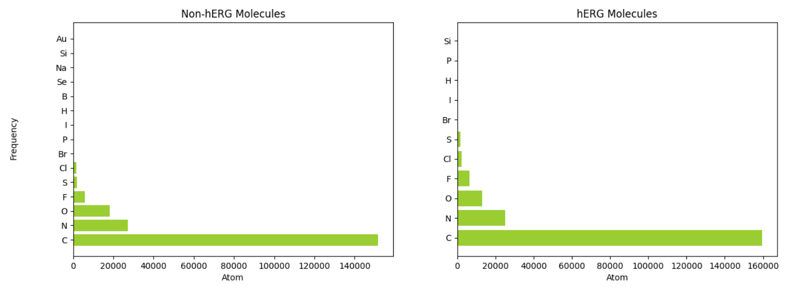


Figure 2: Atom Distribution for Each Class

Examining the results in Table 1, it is clear that the molecular level features are nearly equivalent between the two classes. Similarly, not one atomic level feature appears to separate the two classes of molecules, as demonstrated by Table 2 and Figure 2. However, the advantage of using graph neural networks is their ability to consider the structure of the molecules as well as the higher-order interactions between the properties within the atoms with the larger molecular structure.

2.3 Dataset Splits

The data was split into 4 sets using a 70%, 10%, 10%, 10% scheme for use in training, validation, conformal calibration, and testing. It is important that each split should exhibit similarly diverse chemical properties. One fear of the train-test-validate data split process with molecules is that the model will only be trained a few molecular structures, so the model will fail to generalize to other structures. One way to assess the structure of a molecule is the Murcko Scaffold, which defines equivalence classes for molecules by reducing each molecule to its core structure. Examples of the Murcko Scaffold are provided in the appendix.

The data is then partitioned so each data split contains equal number of elements from each class in the Murcko Scaffold. This ensures that the data the model is trained, validated, calibrated, and tested on exhibits similar molecular properties to the wide range of structures that exist in nature. In addition, it is important to assess the distribution of hERG and non-hERG blocking molecules within each split. Particularly for the training set, it is important that the model does not overfit to one particular class over another. As shown in Figure A1, after assessing the class balances found in split, the classes are well balanced.

3 Methods

3.1 Graph neural networks

This project explores the affect of various graph neural network layer choices on the predictive power of a hERG blocking model. One advantage of using such layers is their ability to handle graphs of different sizes. This is particularly attractive for this application, as the molecules in the dataset contain various numbers of atoms, and thus produce different size graphs. The three layers defined below can handle graphs with different number of nodes, as long as the feature vector for each node is of the same length. The three GNN layers utilized in this project are two different graph convolutional layers as well as a graph attention layer.

Graph Convolutional Layer 1 (GCN) First, the graph convolutional layer from Kipf and Welling [6] was implemented. This layer will be referred to as "GCN", to keep with the naming convention of the Pytorch Geometric package. This particular operator outputs

$$X' = \hat{D}^{-1/2}\hat{A}\hat{D}^{-1/2}X\Theta \tag{1}$$

where $\hat{A} = A + I$ provides the adjacency matrix with self-loops and $\hat{D}_{ii} = \sum_{j=0} \hat{A}_{ij}$ is the diagonal degree matrix, and Θ are the learnable weights

Graph Convolutional Layer 2 (GraphConv) However, that is not the only option for message passing with graph convoluions. Again, this will be referenced as "GraphConv" to reflect the name of its implementation in Pytorch Geometric. The formulation from from Morris et al. [5].

$$x_i' = W_1 x_i + W_2 \sum_{j \in \mathcal{N}(i)} e_{j,i} \cdot x_j \tag{2}$$

where $e_{j,i}$ denotes the edge weight from source node j to target node i. For this project, edge weights were considered to be 1.

Graph Attention Layer Similar to computer vision neural networks, convolution is not the only method for handling the graph structures. Veličković et al. [7] develops the idea of attention for graph neural networks in particular. This is implemented as

Table 3: Training Hyperparameters

Parameter	Value
Batch Size	100
Learning Rate	0.0001
Weight Decay	0.001

$$x'_i = \alpha_{i,j}\Theta x_i + \sum_{j\in\mathcal{N}(i)} \alpha_{i,j}\Theta x_j \tag{3}$$

where the coefficients are computed with

$$a_{i,j} = \frac{exp(LeakyReLU(a^T[\Theta x_i||\Theta x_j]))}{\sum_{k\in\mathcal{N}(i)(i)} exp(LeakyReLU(a^T[\Theta x_i||\Theta x_k])} \tag{4}$$

3.2 Model training and implementation

The same general architecture was implemented for each of the three models, with the only difference being the GNN layer used within the model. Each model consists of 3 GNN layers with a ReLU activation function following each layer. The Graph Attention Layer also contains 3 attention heads. As depicted in the schematic in Figure A2, the first GNN layer has 79 input channels, to accommodate the length of the atomic feature vector, and then outputs 1024 channels. The next 2 hidden GNN layers both have 1024 input and output features. The output of the final GNN layer is then fed into a global max pooling function, before it is fed into a final dense linear layer that outputs to a single node. The sigmoid activation function is then applied to map this output into a score between 0 and 1. All of these models were implemented using the Pytorch Geometric package. In the early developmental stages of this model, it was found the model was quick to over fit to the training set. This emphasized the need for regularization and motivated the choice to include 10% dropout in each of the GNN layers, and 50% dropout in the final linear layer.

Each model was trained using the Adam optimizer for 100 epochs. Various techniques to provide regularization to the models were also implemented during the training process, including increasing the batch size and adding weight decay to the optimizer. A grid search over the combinations of various hyperparameters resulted in the final hyperparameters selected for the model, presented in Table 1.

Since the output of the model is binary, the model was trained with Binary Cross Entropy loss. Area under the precision recall curve was utilized to assess the model's performance as it trains. Since the threshold value, λ, to classify a given output score as hERG blocking or not is defined later using conformal risk control, metrics such as accuracy and F1 scores are not applicable during the model training process.

3.2.1 Conformal risk control

Although neural networks often provide improved prediction performance, they do not provide a straight-forward method for inference on their predictions. In the medical setting, it is vital to understand uncertainty quantification in order to avoid model failure. Conformal prediction and risk control provide a solution to this dichotomy. The conformal framework produces statistically rigorous uncertainty sets and intervals, without the need for distribution or model assumptions. Specifically, for any model $\hat{f}$, and any notion of model uncertainty, rigorous prediction sets or risk control can be conducted.

This procedure requires data that is not seen by the model during the training process, called the calibration set. This data is used to calibrate the procedure and find the parameter of interest such that the prediction sets or threshold value can be found.

For this particular problem, conformal risk control provides a framework for selecting a threshold $\hat{\lambda}$ for classifying the outputs of the model as hERG blockers or not. Anastasios et al. [2] develops the following framework to provide such guarantees. Conformal risk control provides guarantees of the form

$$\mathbb{E}(\ell(\mathcal{C}_{\hat{\lambda}}(X_{test}), y_{test})) \leq \alpha \tag{5}$$

where ℓ is a non-increasing function, $\mathcal{C}_{\hat{\lambda}}(\cdot)$ is the model's prediction using the threshold λ, and α is the pre-determined rate you want to control.

If the loss function, $\ell(\mathcal{C}_{\lambda}(x), y) \in (-\infty, B]$ for $B < \infty$, $\hat{\lambda}$ is found by the following:

$$\hat{\lambda} = inf\{\lambda : \hat{R}(\lambda) \leq \alpha - \frac{B - \alpha}{n}\} \tag{6}$$

Where n is the size of the calibration set and $\hat{R}(\lambda) = (\ell(\mathcal{C}_{\hat{\lambda}}(x_1), y_1) + \cdots + (\mathcal{C}_{\hat{\lambda}}(x_n), y_n))/n$ is the empirical risk found using the calibration set.

In this particular application, the loss function is the false negative rate, since we want to limit the number of incorrectly classified hERG blocking molecules, both for the health of patients down the line, and for the improvement of drug development. Hence $B = 1$, and

$$\ell = \frac{\sum \mathbb{I}(\hat{f}(x_i) = 0 \cap \hat{f}(x_i) \neq y_i)}{\sum \mathbb{I}((\hat{f}(x_i) = 0 \cap \hat{f}(x_i) \neq y_i) \cup (\hat{f}(x_i) = 0 \cap \hat{f}(x_i) = y_i))} \tag{7}$$

is the false negative rate.

Although the conformal risk control framework is the most appropriate for this application, the theory of conformal inference will be utilized to validate that the size of the calibration set is appropriate for the application. Instead of considering the value of $\hat{\lambda}$ that guarantees that the false negative rate is less than α, a set of labels will be will be found with the guarantee that the true label is within the set with probability very near $1 - \alpha$. Mathematically, this is expressed as:

$$P(Y_{test} \in \mathcal{C}(X_{test})) \geq 1 - \alpha \tag{8}$$

A natural link exists between the results in Equations 5 and 8: the result shown in Equation 8 can be established if the the loss function, ℓ, is $\ell(C(X_{test}), Y_{test}) = \mathbb{I}\{Y_{test} \notin C(X_{test})\}$. Switching the loss function allows for the distribution of the prediction sets to be expressed as:

$$\mathcal{P}(Y_{test} \in \mathcal{C}(X_{test})|\{X_i, Y_i\}_{i=1}^{n}) \sim Beta(n + 1 - \ell, \ell) \tag{9}$$

where $\ell = (n + 1)\alpha$, as shown in Vovk [7].

Using this distribution, it is then possible to calculate the ϵ with n points such that the coverage of $1 - \alpha \pm \epsilon$ is probability $1 - \delta$. It is important to note that coverage is always at least $1 - \alpha$: δ controls the tail probability of the coverage conditional on the calibration set.

For this particular application, α was chosen to be 0.1, δ was chosen to be 0.1, and ϵ was chosen as 0.013935, since the calibration set was set as 10% of the total dataset, resulting in 1300 molecules. Hence, the coverage of the prediction sets is $90\% \pm 1.4\%$ with probability of 90%.

The theoretical result presented in Equation 9 also allows for the empirical distribution of the coverage to be compared with the expected theoretical distribution. In order to estimate this distribution of the coverage, new validation and calibration datasets would be sampled R times, with the conformal procedure calculated for each. Since the size of the dataset is finite, new calibration and validation sets cannot be generated. Instead, the calibration and validation data splits are combined and then randomly shuffled R times. The conformal procedure is then calculated for each new split to create the distribution of the coverage. One would expect the mean of the distribution to be exactly $1 - \alpha$ and the standard deviations to be similar to those from the expected Beta distribution, although slight deviations may occur from the finiteness of the datasets.

In the case of binary classification, prediction sets are not as interesting as controlling the false negative rate. However, this transition allows for the validation of the size of the calibration set, which is vital to guarantee if the conformal risk control procedure is working.

4 Results

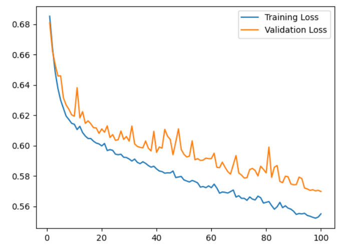
4.1 Model Training

As demonstrated by the curves plotted in Figure 5, the model implemented with the second graph convolutional layer, GCN, resulted in the best training metrics. GraphConv performed the next best, with the graph attention network, GAN, performing the worst of all three models. However, it is also clear that the model with the GraphConv layers is overfitting to the training set. Further regularization is needed to mitigate the overfitting and boost the performance of the model on the validation set.

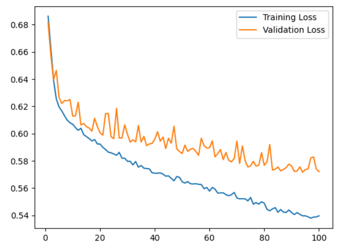
Table 4: Number of Trainable Parameters

Model	$\hat{\lambda}$
GCN	1132545
GAT	1136641
GraphConv	2262017

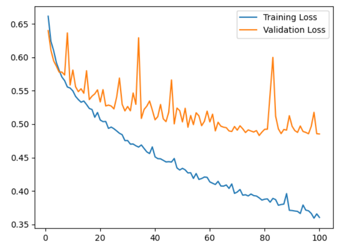
It is also evident that the GCN model is not overfitting to the training set. Training the GCN model for longer could also potentially improve the performance of the model on the validation set. As shown in Table 4, GraphConv was the most complex model implemented with nearly double the number of trainable parameters of the other two models. Hence, in this application, a more complex model is a better for approximating the relationship between molecules and their hERG blocking properties.



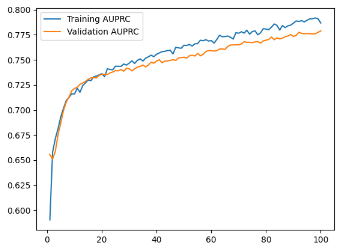
(a) GCN Loss



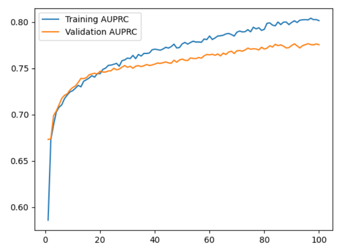
(b) GAT Loss



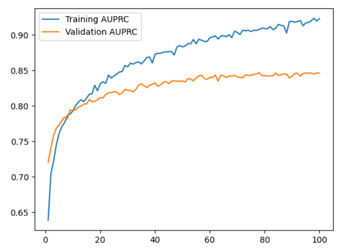
(c) GraphConv Loss



(d) GCN AUPRC



(e) GAT AUPRC



(f) GraphConv AUPRC

Figure 3: Plots of training and validation loss and AUPRC for each model over the 100 epochs.

4.2 Conformal Risk Control

After performing conformal risk control, the following values of $\hat{\lambda}$ were chosen such that the probability of the false negative rate is less than 10% with probability 90%.

Table 5: Estimated Threshold Parameter

Model	$\hat{\lambda}$
GCN	0.3412
GAT	0.3421
GraphConv	0.4647

As would be expected, the best performing model has the highest threshold parameter. A larger value of area under the precision recall curve implies the false negative rate will be smaller. Hence a larger threshold will still comply with the false negative rate restriction. Since the model with the GraphConv layers provided the highest area under the precision recall curve, it then follows that the GraphConv model would have the highest threshold parameter selected by the conformal risk control procedure.

In addition, it is also necessary to validate that the size of the calibration set is sufficient for this application. To do this, the empirical distribution of the coverage of the prediction intervals is compared to the theoretical distribution, as expressed in Equation 9. As shown in Figure 6, the distributions of the coverage coincide with the theoretical expectations of the corresponding Beta distributions. The mean values of the empirical distributions are extremely close to 0.9, while the empirical standard deviations for the $5th$ and $95th$ quantiles are also similar to the quantiles for a $Beta(n + 1 - \ell, \ell)$ distribution. This process validates that the conformal procedure is working, and confidence can be placed in the values selected for the threshold parameter.

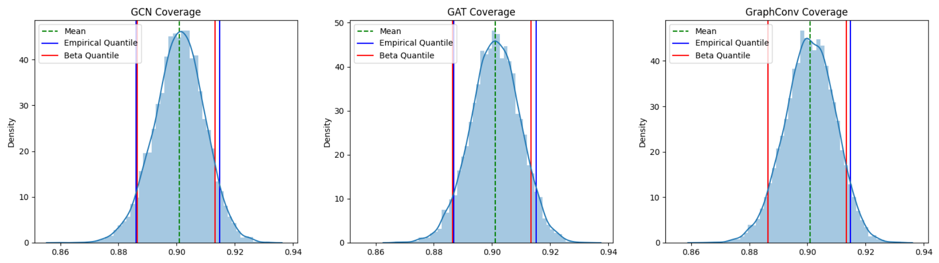


Figure 4: Evaluating the coverage provided by the conformal risk control procedure.

Using the threshold values found with conformal risk control, the three models can then be compared using accuracy, precision, recall, and F1 scores.

Table 6: Test Set Metrics

Model	FNR	Accuracy	Precision	F1
GCN	0.0993	67.08 %	0.6114	0.7284
GAT	0.0993	67.05 %	0.6112	0.7282
GraphConv	0.0993	82.37 %	0.7896	0.8415

As a sanity check, the false negative rate was calculated for the test set. It may appear odd that the false negative rate for the test set is not 0.1. However, as demonstrated in Equation 6, the conformal risk control procedure is calibrated to produce a slightly more conservative level than α. Precision and F1 were chosen as metrics since precision gives insight into the false positives the model is producing, while F1 reflects a balance between precision and recall. Since the false negative rate is being controlled by the conformal risk control procedure, the models are producing more false positives, so the precision is lower than the F1 score. Evaluating the other metrics, it is again clear that the GraphConv layer model provides the best all-around performance. While the accuracy for this model is the highest of the three models, the GraphConv model also performs the best accross the other metrics. The precision is 17% higher and the F1 score is 11% higher than the other models.

The GCN and GAT models perform very similar in terms of testing metrics. Hence, the most complex model of the three, the GraphConv model, is certainly the best for prediction molecules as hERG blocking.

5 Conclusion

Conformal risk control allows for probabilistic guarantees to be applied to the outputs of neural networks. The combination of these methods allows for procedures that both leverage the predictive power and flexibility of neural networks while simultaneously guaranteeing the quality of such predictions. This framework does come with the price of requiring more data to calibrate the conformal procedure. In a framework where we already allocate potential training data to validation and testing, removing more data to create a further data split could be seen as unnecessarily wasteful. However, in medical applications such as this one, the benefits of controlling the false negative rate far outweigh the cost of losing data points from one of the other data splits.

The importance of the calibration set on the output of the values in the conformal prediction procedure does bring rise to an important point– each data split needs to reflect the true underlying distribution of the data generating process. In the case of molecule prediction, it is important to consider how certain molecule structures are divided each of the data splits. The benefits of this consideration were confirmed by the success of the coverage values in the conformal risk control procedure.

The best performing model with an accuracy of 82.37% and F1 score of 0.84 is in line with the current implementations of graph neural networks presented in Karim et al. [4]. Evidently though, there is still a lot of room for improvement. Further research is needed to verify if other model architectures would increase model performance, or if intelligent feature engineering of atom and bond features would also boost model performance.

Ultimately, graph convolutional neural networks provide a powerful architecture to classify molecules as hERG blocking molecules or not. Conformal risk control then allows for the false negative rate of these predictions to be controlled. In practice, this model could be used to confidently rule out molecules that are not hERG blocking, while identifying molecules that should be re-examined and tested by domain experts.

References

References follow the acknowledgments in the camera-ready paper. Use unnumbered first-level heading for the references. Any choice of citation style is acceptable as long as you are consistent. It is permissible to reduce the font size to `small` (9 point) when listing the references. Note that the Reference section does not count towards the page limit.

[1] Ahmad W, Tayara H, Chong KT. Attention-Based Graph Neural Network for Molecular Solubility Prediction. ACS Omega. 2023 Jan 12;8(3):3236-3244. doi: 10.1021/acsomega.2c06702. PMID: 36713733; PMCID: PMC9878542.

[2] Anastasios N. Angelopoulos, Stephen Bates. (2022). A Gentle Introduction to Conformal Prediction and Distribution-Free Uncertainty Quantification.

[3] Huang, K., Fu, T., Gao, W., Zhao, Y., Roohani, Y., Leskovec, J., Coley, C., Xiao, C., Sun, J., Zitnik, M. (2021). Therapeutics Data Commons: Machine Learning Datasets and Tasks for Drug Discovery and Development. Proceedings of Neural Information Processing Systems, NeurIPS Datasets and Benchmarks.

[4] Karim, A., Lee, M., Balle, T. et al. CardioTox net: a robust predictor for hERG channel blockade based on deep learning meta-feature ensembles. J Cheminform 13, 60 (2021). https://doi.org/10.1186/s13321-021-00541-z

[5] Morris, C., Ritzert, M., Fey, M., Hamilton, W. L., Lenssen, J. E., Rattan, G., Grohe, M. (2019). Weisfeiler and Leman Go Neural: Higher-Order Graph Neural Networks. Proceedings of the AAAI Conference on Artificial Intelligence, 33(01), 4602-4609. https://doi.org/10.1609/aaai.v33i01.33014602

[6] Thomas N. Kipf, Max Welling. (2017). Semi-Supervised Classification with Graph Convolutional Networks.

[7] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, Yoshua Bengio. (2018). Graph Attention Networks.

[8] V. Vovk, “Conditional validity of inductive conformal predictors,” in Proceedings of the Asian Conference on Machine Learning, vol. 25, 2012, pp. 475–490.

6 Appendix

6.1 Video Link

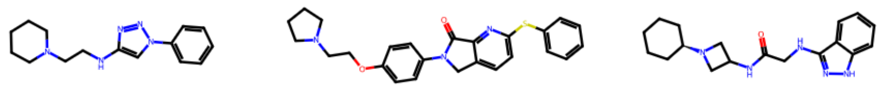
The video for the presentation can be found at: `https://drive.google.com/file/d/1YFUpVSdIU8NapKIr-ZHofBtj_S-AzONn/view?usp=sharing`

6.2 Code

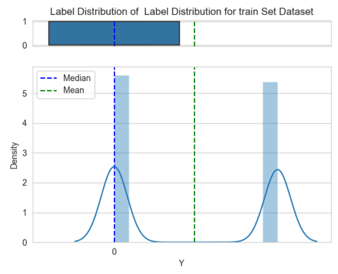
The code for this experiment can be found at: https://github.com/kaitsmith22/hERG$_{b}lockers$

6.3 Murcko Scaffolds

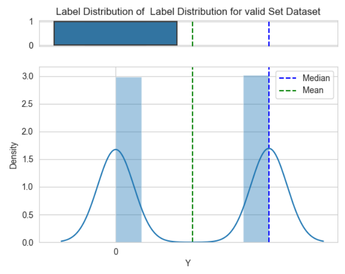
Bellow are examples of the Murcko scaffolds of the molecules from TDC dataset shown in Figure 1. Notice that the Murcko scaffolds of the molecules are less complex than the full molecular structure, as presented in Figure 1. These simplified structures represent core molecular structures that different molecules the in the dataset will share.



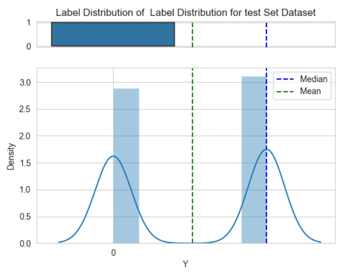
6.4 Label Distributions in Class Splits



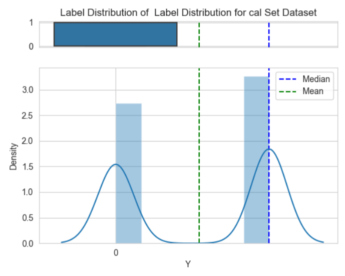
(a) Training Set



(b) Validation Set



(c) Testing Set



(d) Calibration Set

Figure 1: Plots of label distribution in each split of the dataset, where the splits were found by enforcing each set has a diverse set of Murcko scaffolds

6.5 Model Implementation

Below a schematic for the model architecture is shown. Note that between each graph layer there is a ReLU activation function, and a sigmoid activation function was used after the final linear layer.

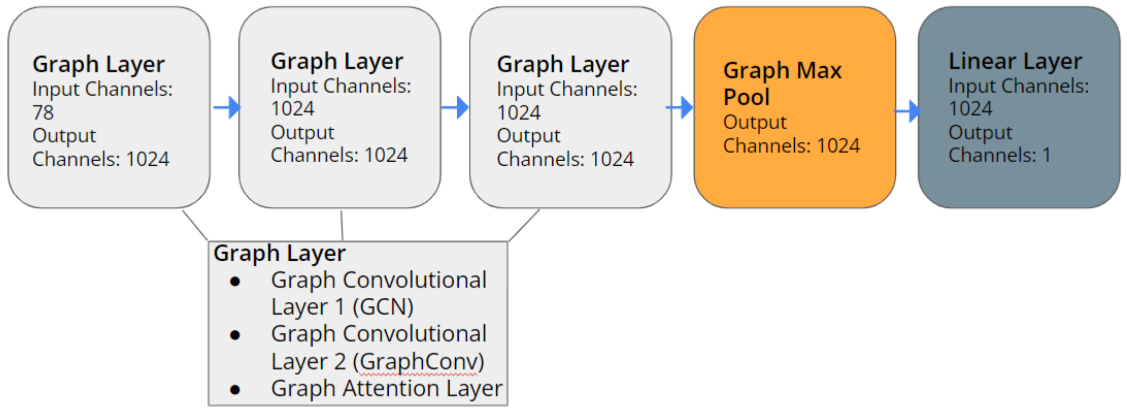


Figure 2: Visualization of model architecture, including the number of channels used in each layer.