MODEL AGNOSTIC GENERATION OF COUNTERFACTUAL EXPLANATIONS FOR MOLECULES

Geemi P Wellawatte
Department of Chemistry
University of Rochester
Rochester, NY, USA

Aditi Seshadri
Department of Chemical Engineering
University of Rochester
Rochester, NY, USA

Andrew D White
Department of Chemical Engineering
University of Rochester
Rochester, NY, USA
andrew.white@rochester.edu

August 13, 2021

ABSTRACT

An outstanding challenge in deep learning is its lack of interpretability. The inability of both designers and users of neural networks to explain *why* a prediction is made is a major drawback. This not only dissuades chemists from using deep learning predictions, but also has led to neural networks learning spurious correlations that are difficult to notice. Counterfactuals are a category of prediction explanations that provide a rationale behind the prediction value. Counterfactuals have satisfying properties like being in the same space as the features and require no deep learning expertise to understand. Counterfactuals are generated via optimization, which makes them complex to use when the features are chemical structures. Here we present an algorithm built on the "Superfast Traversal, Optimization, Novelty, Exploration and Discovery" method (Nigam et. al, 2021) that is efficient and model-agnostic for generating counterfactuals. We show this method is applicable in random forest models, sequence models, and graph neural networks in both classification and regression. Our method requires no additional training, data, or model gradients.

1 Introduction

Deep learning has made significant impacts in chemistry because of its ability to regress non-linear relationships between structure and function.[1] Applications vary from computing quantum properties[2,3] to predicting chemical properties[4,5] to screening drug molecules.[6,7] More specifically, deep neural networks that take in raw graph representations of molecules have proven to be successful when compared with counterparts based on fixed descriptors in both regression and classification tasks.[8] Despite their empirical accuracy, neural networks are black-box models; they lack interpretability and predictions come without explanation.

Explainable artificial intelligence (XAI) is an emerging field which aims to provide explanations, interpretation, and justification for model predictions. XAI should be a normal part of the AI model lifecycle. It can identify data bias and model fairness.[9] Users are more likely to trust and use a prediction if it has an explanation.[10] Finally, it is becoming a legal requirement in some jurisdictions for AI to provide an explanation when used commercially.[11,12] From a researcher's perspective, XAI can also find the so-called "Clever Hans" effects whereby a model has learned spurious correlations such as the existence of a watermark in images or an over representation of counterions in positive molecule examples.[13] Despite these benefits of XAI, this is rarely a part of deep learning in chemistry.

Miller[14] proposes a nomenclature within XAI that distinguishes between a prediction *explanation*, *interpretability* of a model, and prediction *justification*. An explanation is a post-hoc description of why a prediction was made by a model.[15] Model interpretability is "the degree to which an observer can understand the cause of a decision."[16] Finally, justification of a prediction is a description of why a prediction should be believed. Justification typically relies on estimated model generalization error. Interpretable models are common in computational chemistry – DFT, molecular dynamics, and linear regression are inherently interpretable models. Justification is also routine, with almost

all recent papers reporting estimated generalization error on withheld test data or from cross-validation. Explanation is rare, especially in deep learning where no insight can be gained by inspecting model weights or parameters.

There are four major approaches for explaining a prediction from a black-box model:[17] identifying which features contribute the most,[18–22] identifying which training data contributes the most,[23] fitting a locally interpretable model around the prediction,[24] and providing contrastive or counterfactual points.[25] Feature importance analysis provides per-feature weights that identify how each feature contributed to the final prediction. These can be formulated as Shapley values,[26] which are a method of computed feature importance weights as a complete explanation (i.e.,$\sum w_i = \hat{f}(x)$).[27] This is effective when working with a sparse set of molecular descriptors, but when working with thousands of descriptors, SMILES or molecular graphs, this can impart little insight to the human understanding.[14] Local interpretable model-agnostic explanations (LIME) provide an implicit "sparsification" relative to feature importance because the locally interpretable model is a different model than the black-box model being explained.[24] For example, a two dimensional linear regression could be the locally interpretable model. The sparsification arises because we can choose the features going into the locally interpretable model and it can be induced by using regularization when fitting the locally interpretable model to the black-box (e.g., using lasso regression).[28] Some care must be taken in choosing the locally interpretable model since it needs to fit well around the prediction and must be specifically constructed for the problem of interest.

Contrastive explanations and counterfactuals explain a prediction by providing related examples of features. These two explanation methods are conceptually similar, but should be distinguished.[25] In contrastive explanations, one tries to answer "why output X, but not output Y?"[29,30] rather than "why did output X happened?". This is similar to generating multiple choice question options after knowing the correct answer. Through the elimination of incorrect options, one is able to recover the reasoning behind the correct answer. On the other hand, counterfactuals try to answer “what is the smallest change to the features that would alter the prediction".[31] A counterfactual is an example as close to the original, but with a different outcome. “Your papers would be better cited, if you had a better title". The example being a paper identical except the new title and the outcome has changed: the paper is better cited. Contrastive explanations generate explanations by entertaining alternate outcomes whereas a counterfactual explanation shows how to minimally modify our input to get a different prediction.

Counterfactuals are intuitive to understand and are sparse because they are as similar to the original prediction as possible.[14,31] Yet counterfactuals are hard to generate because they arise from optimization over input features – which requires special care for molecular graphs.[32,33] Namely, molecular graphs are discrete and have valency constraints, making gradients intractable for computation. Here we propose a method that can generate *molecular* counterfactuals for arbitrary models. These molecular coutnerfactuals provide explanations that are sparse and composed of molecular structures.

An example of a molecular counterfactual is shown in Figure 1. The left molecule is inactive and the right is active. It shows that the carboxylic acid could be made an ester to change activity, giving insight into the reason why the left molecule is not active. The explanation is sparse and intuitive to those with a knowledge of chemical structures. A related concept analogous to counterfactuals is the idea of paired molecules,[34] where similar molecules with opposite activity are used to understand a class of active compounds.

Our approach to generating molecular counterfactuals is built on the Superfast Traversal, Optimization, Novelty, Exploration and Discovery (STONED) method which enables rapid exploration of chemical space without a pre-trained generative model or set of reaction rules.[35] We expand chemical space around the molecule being predicted (base), identify similar molecules with a changed prediction (counterfactuals), and select a small number of these molecular counterfactuals with clustering/Tanimoto similarity. This method works because we represent molecules as Self-Referencing Embedded Strings and any modification to a SELFIES is also a valid molecule.[36] An overview of this process is shown in Figure 2.

1.1 Comparison to existing work

Recent progress in applying XAI methods to graphs, like molecular graphs, is reviewed in Yuan et al.[37] Our method, called Model Agnostic Counterfactual Compounds with STONED (MACCS), produces counterfactual explanations. Counterfactuals are challenging due to the numerical problems associated with both neural networks gradients and working with graphs GNNs.[38] There have been a few counterfactual generation methods for GNNs. The Counterfactuals-GNNExplanier from Lucic et al. uses graph edge operations and a relaxed model prediction function to propose counterfactuals and was found to do well on graph datasets.[32] Graph edge operations cannot be used on molecular structures because the majority of graph operations will violate valencies. This method also requires model gradients with respect to input, which may not be possible for models outside of neural networks. Our method works on

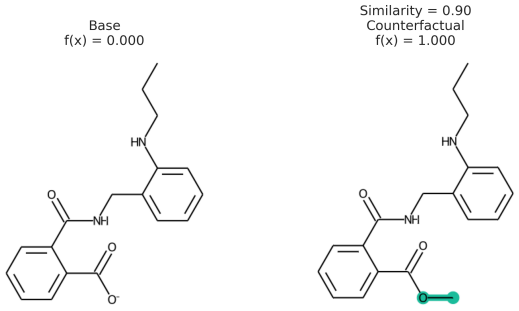

Figure 1: An example of a counterfactual. The molecule on left was predicted to have class of 0, no activity. With the modification shown in teal, the molecule would be in class 1, active. This shows that the carboxylic acid is an explanation for lack of activity.

descriptors, graphs, SMILES, and SELFIES features. MACCS does not require gradients, enabling its use on machine learning methods like random forest classification or support vector machines.

Numeroso et al.[33] proposed a molecular explanation generator that is closer to our work. They use a reinforcement learning agent to generate counterfactuals, which ensures that proposed counterfactuals are reasonable molecules. Our method does not require training a counterfactual generator because all molecules resulting from STONED are valid compounds.[35] This negates the need for a generative counterfactual maker and greatly simplifies the method.

2 Theory

A deep learning model takes in as input a set of feature vectors (x), and outputs a prediction, denoted as $\hat{f}(x)$ or $\hat{y}$. The true value of the property being predicted by the model is denoted as $f(x)$, or y. For chemical applications, x is typically a representation of a molecule, which can be a string (SMILES or SELFIES), a set of chemical descriptors, or a molecular graph. Programs including Mordred[39] and DRAGON[40] can be used to compute chemical descriptors, such as electronegativity or molecular weight, for each molecule. A molecular graph can consist of a node feature vector and an adjacency matrix. The node feature vector provides information on the type of atoms (e.g., C, H, O, N) present in the molecule and the adjacency matrix provides information on the edges between each node, or which atoms are bonded together.[1] Together, the node feature vector and adjacency matrix can be used as a molecular graph input to a graph neural network model.[41]

A counterfactual x' is specific to the example of interest x, where we have made a prediction $\hat{f}(x)$. The counterfactual is the explanation of x and defined by the solution to the following constrained optimization:problem[31]

$$\begin{aligned} \text{minimize} \quad & d(x, x') \\ \text{such that} \quad & \hat{f}(x) \neq \hat{f}(x') \end{aligned} \tag{1}$$

where x is the feature vector of our prediction, $d(x, x')$ is a measure of distance between features, and $\hat{f}(x)$ is our model. The counterfactual optimization problem is a function of x, so that each time a new prediction is made the counterfactual is also updated. In this work, distance is computed with Tanimoto similarity of molecular fingerprints.[42]

In principle, this optimization problem could be solved by computing a gradient $\nabla_x \hat{f}(x)$. However, there are complexities of computing gradients with respect to x because it may be a molecular graph, a SMILES string, or descriptors which then propagate derivatives to the molecular structure (although see recent progress specifically with SELFIES[43,44]). Instead, previous for counterfactual generation have relied on perturbing x using graph transformation operators[32] and reinforcement learning.[33] Both these methods have the disadvantage that they can generate chemically infeasible structures, although Numeroso et al.[33] can generate good candidate molecules with sufficient training. Our innovation here is to use the STONED SELFIES method[35] which rapidly explores local chemical space around a point by exploiting the surjective property of SELFIES (SELF-referencIng Embedded Strings): every SELFIES string is a valid molecule. Krenn *et al.*[36] introduced SELFIES to overcome one of the major limitations in SMILES[45] that, they do not always correspond to valid molecules. The STONED protocol consists of string insertion, deletion, and modification steps that can generate thousands of perturbations of x that are valid molecules and close in chemical space. This requires no training, is independent of features (e.g., molecular graphs, SMILES, descriptors), and requires no gradients.

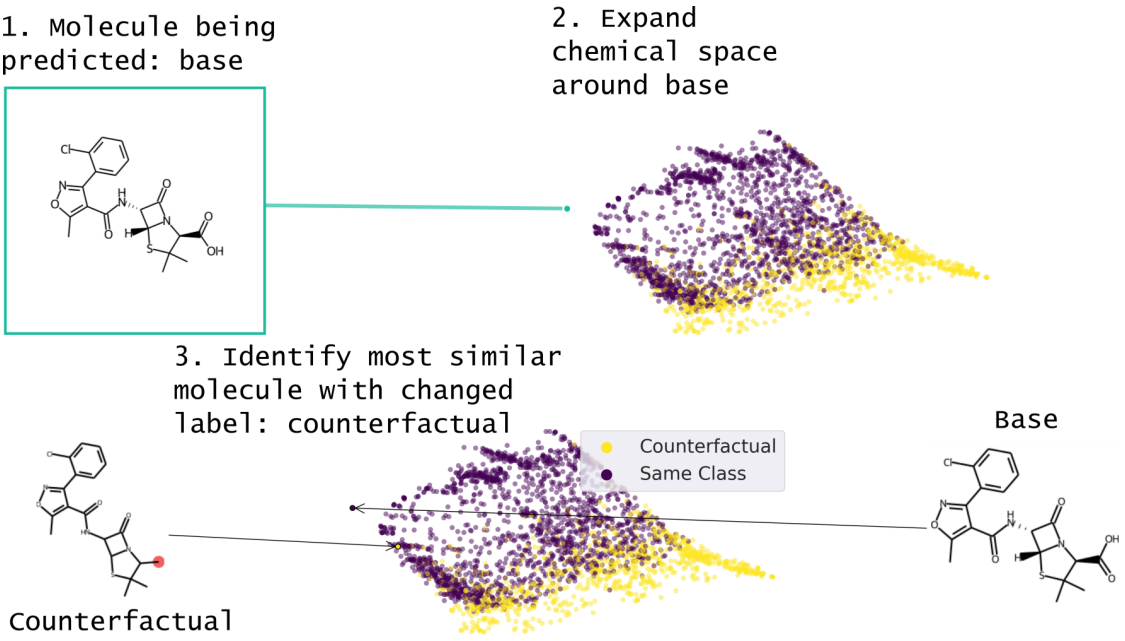

Figure 2: Overview of MACCS. The input is a molecule to be predicted. Chemical space is expanded and clustered. Counterfactuals are selected from clusters to find succinct explanation of base molecule prediction.

3 Methods

An overview of our method is shown in the schematic in Figure 2. We use the STONED method as described in Nigam et al.[35] to sample chemical space. Briefly, a starting molecule is encoded into SELFIES and successive rounds of token deletion, replacement, and insertion is done to generate modifications of the starting molecule. This process relies on the surjective property of SELFIES. As in Nigam et al., we limit the number of modifications to the starting SELFIES to ensure we stay local in chemical space. Additionally, starting diversity is improved by exploiting the fact there are multiple non-canonical starting SELFIES. Unless otherwise stated, 3,000 modified SELFIES are generated with at most 2 token modifications (mutations). The available tokens (alphabet) for insertion/modification in the STONED algorithm are modified here to use a restricted subset of "intuitive" tokens. Specifically, all positively and negatively charged atoms except O^- were removed and the available elements were restricted to B, C, N, O, S, F, Cl, Br, I. We call this the “basic" alphabet. This alphabet can be modified and is discussed further in the results.

Rdkit was used for molecule processing, including constructing molecular graphs, drawing molecules, validating input structures, and computing fingerprints.[46] The scores used in STONED were the Tanimoto similarity[42] of EFPC4[47] fingerprints.

STONED generates a set of molecules around the molecule from which we are predicting (base molecule). To generate counterfactuals, we apply the optimum condition in Equation 1. To generate multiple counterfactuals, clustering is done using DBSCAN[48] with parameters $\epsilon = 0.15$ and minimum 5 samples per cluster. The distances used for clustering $d = 1 - s$, where s is pairwise Tanimoto similarity. The most similar molecule from each cluster which satisfies the counterfactual condition is selected and a further reduction by similarity is done if fewer counterfactuals are requested than clusters. DBSCAN infers cluster numbers using the $\epsilon = 0.15$ parameter, which is in units of similarity.

4 Experiments

4.1 Blood-Brain Barrier Permeation Prediction

Predicting if a molecule can permeate the blood-brain barrier is a classic problem in computational chemistry.[49] The most used dataset comes from Martins et al.[50] It is a binary classification problem with molecular structure as the features. State-of-the-art performance is 0.955–0.988 receiver-operator characteristic area under curve (ROC-AUC) depending on model type and molecular structure featurization.[49] To test MACCS on this dataset, we developed a random forest model as implemented in Scikit-learn[51] using molecular descriptors as features. The descriptors are

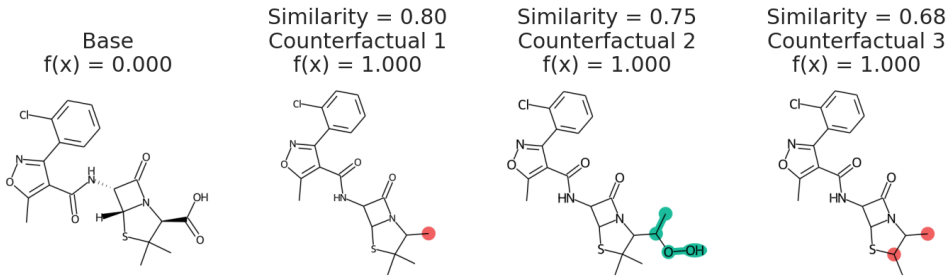

Figure 3: Counterfactual for negative example of Blood-Brain Barrier random forest model. Similarity is computed from Tanimoto similarity of ECFP4 fingerprints.[47] Red indicates deletion relative to base molecule and teal indicates modification. Counterfactuals show that the removing or modificting carboxylic acid group is the simplest way to make this molecule pass the blood-brain barrier.

computed with Morderd.[39] A 20% train/test split was done and the ROC-AUC was computed as 0.91 (See Figure S1 for ROC curve).

Figure 3 shows a negative prediction from the trained blood-brain barrier classifier. The molecule should not pass the blood-brain barrier. The counterfactuals show what could make the negative example cross the blood-brain barrier, including removing the carboxylic acid (Counterfactual 1,3) or changing to an alcohol with additional alkane chains (Counterfactual 2). Based on these counterfactuals, the explanation of why this molecule cannot cross the blood brain barrier is due to the carboxylic acid group. In words: "This molecule will not cross the blood-brain barrier. It would cross the blood-brain barrier if the carboxylic acid were removed."

4.2 Small Molecule Solubility Prediction

Solubility in water plays a critical role in drug design.[52] Thus, there are many previously developed machine learning tools[32,53,54] to predict solubility. Solubility is also an intuitive concept that is taught in introductory organic chemistry, thus providing a good setting to test MACCS. We used solubility data from Sorkun et al.[55], which consists of organic and organometallic molecules. Solubility of the molecule in water is measured in log molarity.

We predict solubility of a given molecule using a gated recurrent unit (GRU) recurrent neural network (RNN)[56] implemented in Keras.[57] RNNs are a standard approach in natural language programming tasks because of their ability to handle long sequences and model long-range correlations. Thus, they are commonly used in chemistry applications with SMILES sequences.[58,59] In our regression model, we use SELFIES because it matches the representation used in MACCS. However, using SELFIES over SMILES does not necessarily translate to better supervised learning performance.[60]

A 10%-10%-80% test-validation-train data split was done. The data, which are specified in SMILES, were canonicalized and converted into SELFIES and training was done for 100 epochs with the Adam optimizer[61] with a learning rate of 10^{-4}. The correlation coefficient on test data is 0.84 and state-of-the-art performance is 0.80–0.93.[62] Additional model details are listed in the Supporting Information.

As this task is regression, Equation 1 must be modified. Instead of finding a change in classification label, we find counterfactuals for both an increase and decrease in solubility:

$$\begin{aligned} &\text{minimize} \quad d(x, x') \\ &\text{such that} \quad \left|\hat{f}(x) - \hat{f}(x')\right| \geq \Delta \end{aligned} \tag{2}$$

where Δ is 1. Figure 4 shows counterfactuals generated for a given base molecule. Increase or decrease in solubility is annotated in the counterfactuals. These counterfactuals can be used to explain what functional groups are most important for solubility of the base molecule. According to Figure 4, the ester, hydrogen bond acceptors, and alkane chain length are contributing reasons for the solubility. The diversity of counterfactuals comes from the DBSCAN clustering, as seen in the principal component analysis projection of chemical space.

4.3 HIV Activity Prediction

Since the first reported case in 1981, the AIDS epidemic has killed 36 million people. According to HIV.gov[63], currently 1.2 million people in the US have tested positive for HIV (human immunodeficiency virus) which causes

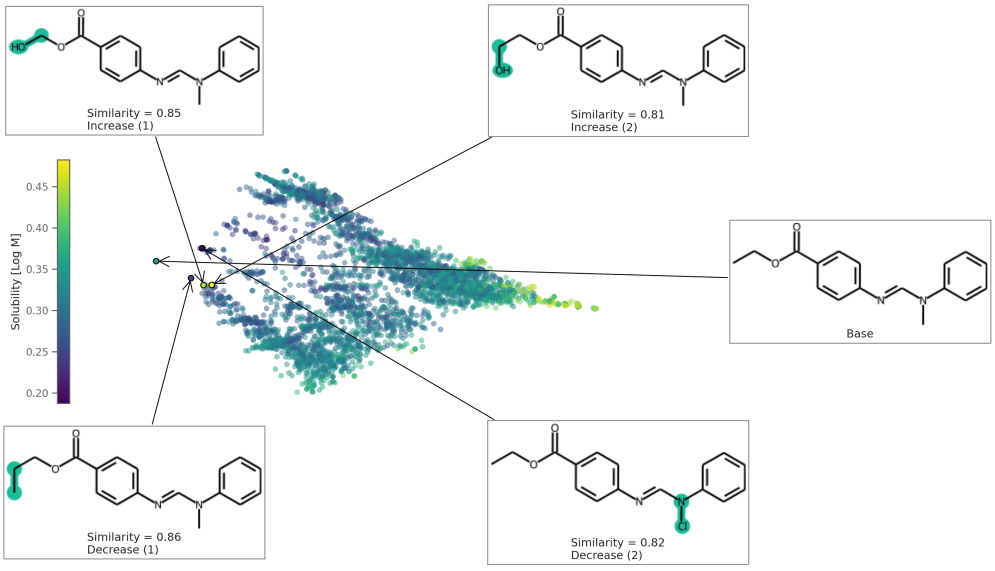

Figure 4: Chemical space for solubility predicting RNN model. This is a principle component analysis of chemical space from Tanimoto similarity distances. Points are colored by solubility. Counterfactuals are annotated.

AIDS. Although there is no cure for HIV, antiretroviral therapy (ART) reduces mortality and transmission of HIV.[64] However, effectiveness of ART is limited due to toxicity and cost of treatment.[65] This means there is still a need for new drugs. Additionally, the National Institute of Allergy and Infectious Diseases has made a systemic study of compounds that can inhibit HIV resulting in large compound datasets. These two facts make predicting potential new HIV drugs a frequently studied task in computational chemistry.[49]

We use a binary classification approach to test MACCS to screen compounds based on their ability to inhibit HIV. The data was downloaded as processed in a Kaggle competition.[66] This dataset was prepared by the Drug Therapeutics Program (DTP) for AIDS antiviral screening for more than 40,000 compounds.[67] We use a graph convolutional network (GCN)[68] implemented in Keras[57] for molecular featurization and standard dense layers for classification based on molecular features. The inputs to this GCN are the molecular graphs generated with canonicalized SMILES using RDKit software.[46] However, in the original dataset only 3.5% of the molecules were labeled HIV active. To address the imbalance between the labels, we used the class weighting technique. A 10%-10%-80% test-validation-train data split was done. The model gains an ROC-AUC of 0.765 after training for only 30 epochs. See Figure S3 for ROC curve. State-of-the-art performance is 0.945–0.993.[69] For more information on this GCN architecture please refer to supplementary information.

After training, we generated counterfactual explanations as shown in Figure 5 for the given base molecule. The base molecule is HIV active. If the base molecule did not have the specific diamide functional group, it would not be active. If the phenyl diether were fluorinated, it would be not active. The fluorination may also be an effect on solubility. Additional counterfactuals are provided in the Figure S4 and reinforce the importance of the diamide group. This shows how chemical reasoning can now be applied to black box predictions through counterfactuals.

4.4 Effect of MACCS Parameters

There are three main parameters to choose in MACCS: the number of molecules to sample, the number of mutations, and the choice of alphabet. The number of molecules to sample is restricted by the speed of inference of the model being evaluated: more is always better. The models from the experiment section are generally fast enough that the majority of time is spent on fingerprint calculation. However, the random forest model is slow to compute descriptors, which is not related to the model or MACCS. Now, we examine the effect of the other two parameters on our RNN model for predicting solubility.

There is no direct relationship between number of SELFIES mutations and the similarity. Figure 6 shows a histogram of molecules arising from STONED as a function of the mutation number from the solubility prediction model. One mutation provides a range of similarities, although few above 0.80 similarity. Even at three mutations, the majority of

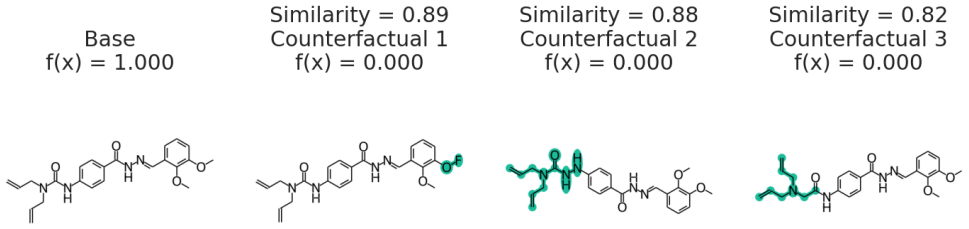

Figure 5: Counterfactuals for positive example of GCN model for classifying HIV activity. Similarity is computed from Tanimoto similarity of ECFP4 fingerprints.[47] Teal indicates the modifications to the base molecule. Counterfactuals illustrate which atom groups make this molecule HIV active.

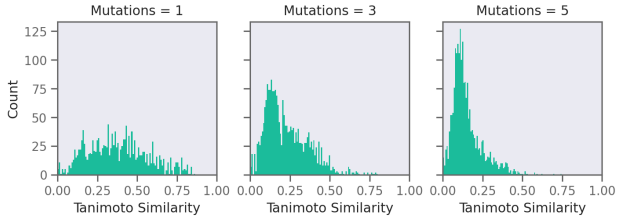

Figure 6: The effect of mutation number on Tanimoto similarity of generated molecules from RNN solubility model. Increasing mutation number reduces number of similar molecules from which counterfactuals can be generated.

molecules are dissimilar and cannot be used for counterfactuals. At five mutations, there are almost no molecules that are comparable with the base molecule. Thus, one and two mutations combined are recommended in MACCS.

The effect of the alphabet choice is shown in Figure 7. Three counterfactuals are shown that are more soluble than the base molecule. In the basic alphabet, recommended for MACCS, we can see that the counterfactual is changing the ester to a ketone. The "Training Data" alphabet is derived from all unique tokens in the training data and shows a counterfactual with a mercury cation. Although technically quite similar, it provides little understanding about why the original molecule is not more soluble. Finally, the SELFIES alphabet without cation/anions removed can propose high similarity counterfactuals simply by ionizing atoms. This does not provide understanding, as these extreme molecules provide little intuition about the base molecule. Although this could be framed as an example of out of distribution predictions, the point of MACCS is to explain predictions and thus we desire an alphabet that results in human interpretable counterfactuals. This is necessarily subjective, but this example shows a limited alphabet provides simpler explanations. Thus, we recmomend the basic alphabet in almost all cases. One exception may be organometallic molecules, where exchanging a metal in a counterfactual may be helpful for understanding.

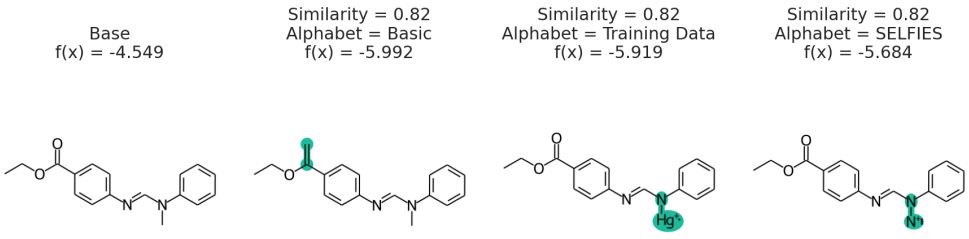

Figure 7: The effect of STONED alphabet choice counterfactuals from RNN solubility model. Although each counterfactual has the same similarity, the molecules are increasingly unusual. The basic alphabet provides a balance of intuitive counterfactuals and enough tokens to explore chemical space.

5 Conclusions

Counterfactuals are human interpretable explanations composed of molecular structures that explain model predictions. Counterfactuals can be generated with our proposed MACCS method. Our method is efficient and general, as demonstrated on three model types and three datasets. It requires no gradients, training, or additional data to generate per-prediction explanations. As deep learning becomes more common in real systems, counterfactual explanations can mitigate model bias by involving experts in understanding predictions. The code for generating counterfactuals available at `https://github.com/ur-whitelab/exmol`.

6 Acknowledgements

This material is based upon work supported by the National Science Foundation under Grant No. 1764415. Research reported in this work was supported by the National Institute of General Medical Sciences of the National Institutes of Health under award number R35GM137966. We thank the Center for Integrated Research Computing (CIRC) at the University of Rochester for providing computational resources and technical support.

References

[1] A. D. White, *Deep Learning for Molecules and Materials*, 2021.

[2] V. L. Deringer, M. A. Caro and G. Csányi, *Advanced Materials*, 2019, **31**, 1902765.

[3] F. A. Faber, L. Hutchison, B. Huang, J. Gilmer, S. S. Schoenholz, G. E. Dahl, O. Vinyals, S. Kearnes, P. F. Riley and O. A. von Lilienfeld, *Journal of Chemical Theory and Computation*, 2017, **13**, 5255–5264.

[4] J. S. Delaney, *Journal of Chemical Information and Computer Sciences*, 2004, **44**, 1000–1005.

[5] A. Lusci, G. Pollastri and P. Baldi, *Journal of Chemical Information and Modeling*, 2013, **53**, 1563–1575.

[6] Z. Xiong, D. Wang, X. Liu, F. Zhong, X. Wan, X. Li, Z. Li, X. Luo, K. Chen, H. Jiang and M. Zheng, *Journal of Medicinal Chemistry*, 2020, **63**, 8749–8760.

[7] R. Huang, M. Xia, S. Sakamuru, J. Zhao, S. A. Shahane, M. Attene-Ramos, T. Zhao, C. P. Austin and A. Simeonov, *Nature Communications*, 2016, **7**,.

[8] D. Jiang, Z. Wu, C.-Y. Hsieh, G. Chen, B. Liao, Z. Wang, C. Shen, D. Cao, J. Wu and T. Hou, *Journal of Cheminformatics*, 2021, **13**,.

[9] F. Doshi-Velez and B. Kim, *arXiv e-prints*, 2017, arXiv–1702.

[10] J. D. Lee and K. A. See, *Human factors*, 2004, **46**, 50–80.

[11] B. Goodman and S. Flaxman, *AI Magazine*, 2017, **38**, 50–57.

[12] *Lex access to European Union law*, `https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX%3A02016R0679-20160504`.

[13] S. Lapuschkin, S. Wäldchen, A. Binder, G. Montavon, W. Samek and K.-R. Müller, *Nature communications*, 2019, **10**, 1–8.

[14] T. Miller, *Artificial intelligence*, 2019, **267**, 1–38.

[15] Z. C. Lipton, *Queue*, 2018, **16**, 31–57.

[16] O. Biran and C. Cotton, IJCAI-17 workshop on explainable AI (XAI), 2017, pp. 8–13.

[17] J. Jiménez-Luna, F. Grisoni and G. Schneider, *Nature Machine Intelligence*, 2020, **2**, 573–584.

[18] M. Sundararajan, A. Taly and Q. Yan, International Conference on Machine Learning, 2017, pp. 3319–3328.

[19] D. Smilkov, N. Thorat, B. Kim, F. Viégas and M. Wattenberg, *arXiv preprint arXiv:1706.03825*, 2017.

[20] G. Montavon, A. Binder, S. Lapuschkin, W. Samek and K.-R. Müller, in *Layer-Wise Relevance Propagation: An Overview*, ed. W. Samek, G. Montavon, A. Vedaldi, L. K. Hansen and K.-R. Muller, Springer International Publishing, Cham, 2019, pp. 193–209.

[21] D. Erhan, Y. Bengio, A. Courville and P. Vincent, *Technical Report, Univeristé de Montréal*, 2009.

[22] J. Jiménez-Luna, M. Skalic, N. Weskamp and G. Schneider, *Journal of Chemical Information and Modeling*, 2021, **61**, 1083–1094.

[23] P. W. Koh and P. Liang, International Conference on Machine Learning, 2017, pp. 1885–1894.

[24] M. Ribeiro, S. Singh and C. Guestrin, Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations, San Diego, California, 2016, pp. 97–101.

[25] A. L. McGill and J. Klein, *Journal of Personality and Social Psychology*, 1993, **64**, 897–905.

[26] L. S. Shapley, *Proceedings of the national academy of sciences*, 1953, **39**, 1095–1100.

[27] S. M. Lundberg and S.-I. Lee, Proceedings of the 31st International Conference on Neural Information Processing Systems, Red Hook, NY, USA, 2017, p. 4768–4777.

[28] F. Santosa and W. W. Symes, *SIAM Journal on Scientific and Statistical Computing*, 1986, **7**, 1307–1330.

[29] I. Stepin, J. M. Alonso, A. Catala and M. Pereira-Fariña, *IEEE Access*, 2021, **9**, 11974–12001.

[30] W. Demopoulos and B. C. van Fraassen, *The Philosophical Review*, 1982, **91**, 603.

[31] S. Wachter, B. Mittelstadt and C. Russell, *Harv. JL & Tech.*, 2017, **31**, 841.

[32] A. Lucic, M. ter Hoeve, G. Tolomei, M. Rijke and F. Silvestri, *ArXiv*, 2021, **abs/2102.03322**,.

[33] D. Numeroso and D. Bacciu, *ArXiv*, 2020, **abs/2011.05134**,.

[34] J. Hussain and C. Rea, *Journal of chemical information and modeling*, 2010, **50**, 339–348.

[35] A. Nigam, R. Pollice, M. Krenn, G. dos Passos Gomes and A. Aspuru-Guzik, *Chemical science*, 2021.

[36] M. Krenn, F. Häse, A. Nigam, P. Friederich and A. Aspuru-Guzik, *Machine Learning: Science and Technology*, 2020, **1**, 045024.

[37] H. Yuan, H. Yu, S. Gui and S. Ji, *arXiv preprint arXiv:2012.15445*, 2020.

[38] D. Balduzzi, M. Frean, L. Leary, J. P. Lewis, K. W.-D. Ma and B. McWilliams, Proceedings of the 34th International Conference on Machine Learning, 2017, pp. 342–350.

[39] H. Moriwaki, Y.-S. Tian, N. Kawashita and T. Takagi, *Journal of cheminformatics*, 2018, **10**, 1–14.

[40] A. Mauri, V. Consonni, M. Pavan and R. Todeschini, *Match*, 2006, **56**, 237–248.

[41] P. W. Battaglia, J. B. Hamrick, V. Bapst, A. Sanchez-Gonzalez, V. Zambaldi, M. Malinowski, A. Tacchetti, D. Raposo, A. Santoro, R. Faulkner *et al.*, *arXiv preprint arXiv:1806.01261*, 2018.

[42] T. T. Tanimoto, *Internal IBM Technical Report*, 1958.

[43] C. Shen, M. Krenn, S. Eppel and A. Aspuru-Guzik, *Machine Learning: Science and Technology*, 2021, **2**, 03LT02.

[44] A. Nigam, R. Pollice and A. Aspuru-Guzik, *JANUS: Parallel Tempered Genetic Algorithm Guided by Deep Neural Networks for Inverse Molecular Design*, 2021.

[45] D. Weininger, *Journal of Chemical Information and Computer Sciences*, 1988, **28**, 31–36.

[46] G. Landrum, *rdkit/rdkit: 2021_03_3 (Q1 2021) Release*, 2021, `https://doi.org/10.5281/zenodo.4973812`.

[47] M. Hassan, R. D. Brown, S. Varma-O'Brien and D. Rogers, *Molecular diversity*, 2006, **10**, 283–299.

[48] M. Ester, H.-P. Kriegel, J. Sander, X. Xu *et al.*, kdd, 1996, pp. 226–231.

[49] Z. Wu, B. Ramsundar, E. N. Feinberg, J. Gomes, C. Geniesse, A. S. Pappu, K. Leswing and V. Pande, *Chemical science*, 2018, **9**, 513–530.

[50] I. F. Martins, A. L. Teixeira, L. Pinheiro and A. O. Falcao, *Journal of chemical information and modeling*, 2012, **52**, 1686–1697.

[51] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot and E. Duchesnay, *Journal of Machine Learning Research*, 2011, **12**, 2825–2830.

[52] K. T. Savjani, A. K. Gajjar and J. K. Savjani, *International Scholarly Research Notices*, 2012, **2012**,.

[53] R. Gozalbes and A. Pineda-Lucena, *Bioorganic & medicinal chemistry*, 2010, **18**, 7078–7084.

[54] W. L. Jorgensen and E. M. Duffy, *Advanced drug delivery reviews*, 2002, **54**, 355–366.

[55] M. C. Sorkun, A. Khetan and S. Er, *Sci. Data*, 2019, **6**, 143.

[56] J. Chung, C. Gulcehre, K. Cho and Y. Bengio, *arXiv preprint arXiv:1412.3555*, 2014.

[57] F. Chollet *et al.*, *Keras*, 2015, `https://github.com/fchollet/keras`.

[58] A. Llinas, I. Oprisiu and A. Avdeef, *Journal of chemical information and modeling*, 2020, **60**, 4791–4803.

[59] J.-C. Bradley, J.-C. Bradley, C. Neylon, R. Guha, A. Williams, B. Hooker, A. Lang, B. Friesen, T. Bohinski, D. Bulger, M. Federici, J. Hale, J. Mancinelli, K. Mirza, M. Moritz, D. Rein, C. Tchakounte and H. Truong, *Nature Precedings*, 2010.

[60] S. Chithrananda, G. Grand and B. Ramsundar, *arXiv preprint arXiv:2010.09885*, 2020.

[61] D. Kingma and J. Ba, *International Conference on Learning Representations*, 2014.

[62] S. Boobier, D. R. J. Hose, A. J. Blacker and B. N. Nguyen, *Nature Communications*, 2020, **11**,.

[63] HIV.gov, *U.S. statistics*, 2021, `https://www.hiv.gov/hiv-basics/overview/data-and-trends/statistics`.

[64] J. A. Sterne, M. A. Hernán, B. Ledergerber, K. Tilling, R. Weber, P. Sendi, M. Rickenbach, J. M. Robins, M. Egger, S. H. C. Study *et al.*, *The Lancet*, 2005, **366**, 378–384.

[65] J. S. Lee, E. Paintsil, V. Gopalakrishnan and M. Ghebremichael, *BMC Medical Research Methodology*, 2019, **19**,.

[66] Elahi, *Cheminformatics*, 2019, `https://www.kaggle.com/mmelahi/cheminformatics/version/4?select=hiv.zip`.

[67] *DTP NCI bulk data for download - Nci DTP data - nci wiki*, `https://wiki.nci.nih.gov/display/NCIDTPdata/`.

[68] T. N. Kipf and M. Welling, International Conference on Learning Representations (ICLR), 2017.

[69] J. Li, D. Cai and X. He, *arXiv preprint arXiv:1709.03741*, 2017.

Supplementary Information

RF model

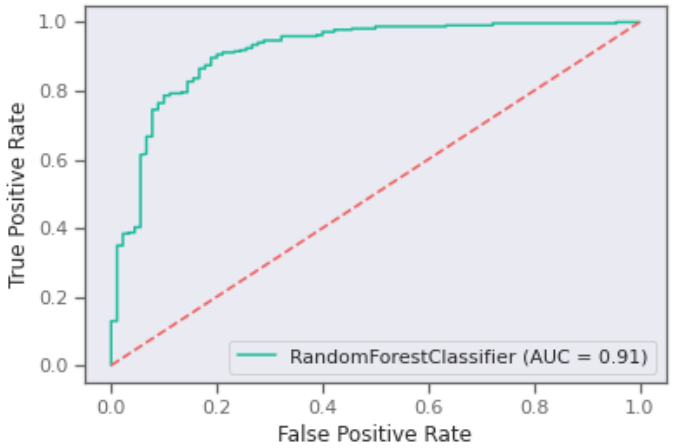

Figure S1: Random forest model fit to test data with area-under curve analysis of receiver operator characteristic.

RNN model

SELFIES tokens are embedded using a 256 dimensional embedding. The embedded sequence is input to a gated recurrent unit (GRU) RNN.[56] The GRU output goes through one hidden dense layer of dimension 128 with an ReLU activation and a solubility is predicted. The loss is mean squared error in units of solubility, log molarity. The Adam optimizer with a learning rate of 0.01 is used in training.[61] Here, N is a variable which refers to the maximum molecule vocabulary length. Sequences are padded with the "[nop]" SELFIES token. The model fit is shown in Figure S2

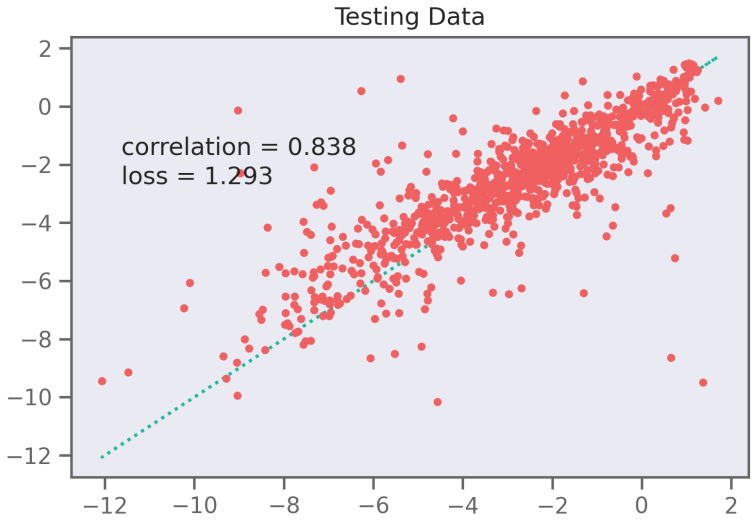

Figure S2: RNN model fit on testing data. Loss is RMSE.

Table SI: RNN model architecture

Layer type	Shape	Activation
Embedding	(N,256)	None
GRU layer	(N,128)	None
Dense layer 1	(128)	ReLU
Dense layer 2	(1)	None

GCN model

Our GCN model follows the original architecture of Kipf and Welling.[68] Namely, our layer definition is:

$$f(V^{(l)}, A) = \sigma\big(\frac{1}{\hat{D}}\hat{A}V^{(l)}W^{(l)}\big) \tag{3}$$

Here, $V^{(l)}$ are the graph level outputs (node features) of layer l. A is the adjacency matrix and $\hat{A} = A + I$ where I is the identity matrix. $\hat{A}$ is used here to add self loops. $\hat{D}$ refers to the node degree matrix. $W^{(l)}$ are the trainable weight matrix for the l^{th} layer.

In our GCN model we stacked 4 graph convolutional layers and 2 dense layers with activation ReLU. The model architecture is shown in table SII. As this is a binary classification task, we use a sigmoid activation in the last dense layer to output predicted HIV activity (HIV inactive: 0 or HIV active: 1). Class weights of 1 and 30 for inactive and active classes are used respectively to address the imbalance in the data. We train our model with binary cross entropy loss and Adam optimizer. A learning rate of 0.01 is used. In this model we have padded the input molecules vectors to be of length 440 (maximum length of molecules in the dataset) to allow batching.

Table SII: GCN model architecture

Layer type	Shape (N=400)	Activation
Input 1	(N,100)	None
Input 2	(N,N)	None
GCN layer 1	(N,100)	ReLU
GCN layer 2	(N,100)	ReLU
GCN layer 3	(N,100)	ReLU
GCN layer 4	(N,100)	ReLU
Graph Reduction layer	(100)	None
Dense layer 1	(256)	Tanh
Dense layer 2	(1)	Sigmoid

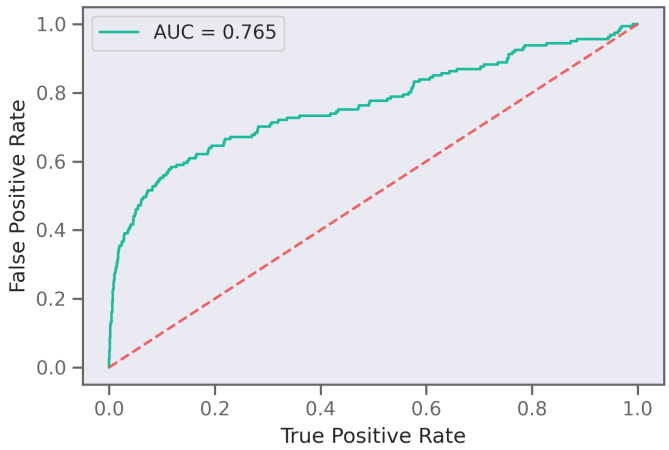

Figure S3: GCN AUC-ROC plot. Loss is binary cross entropy.

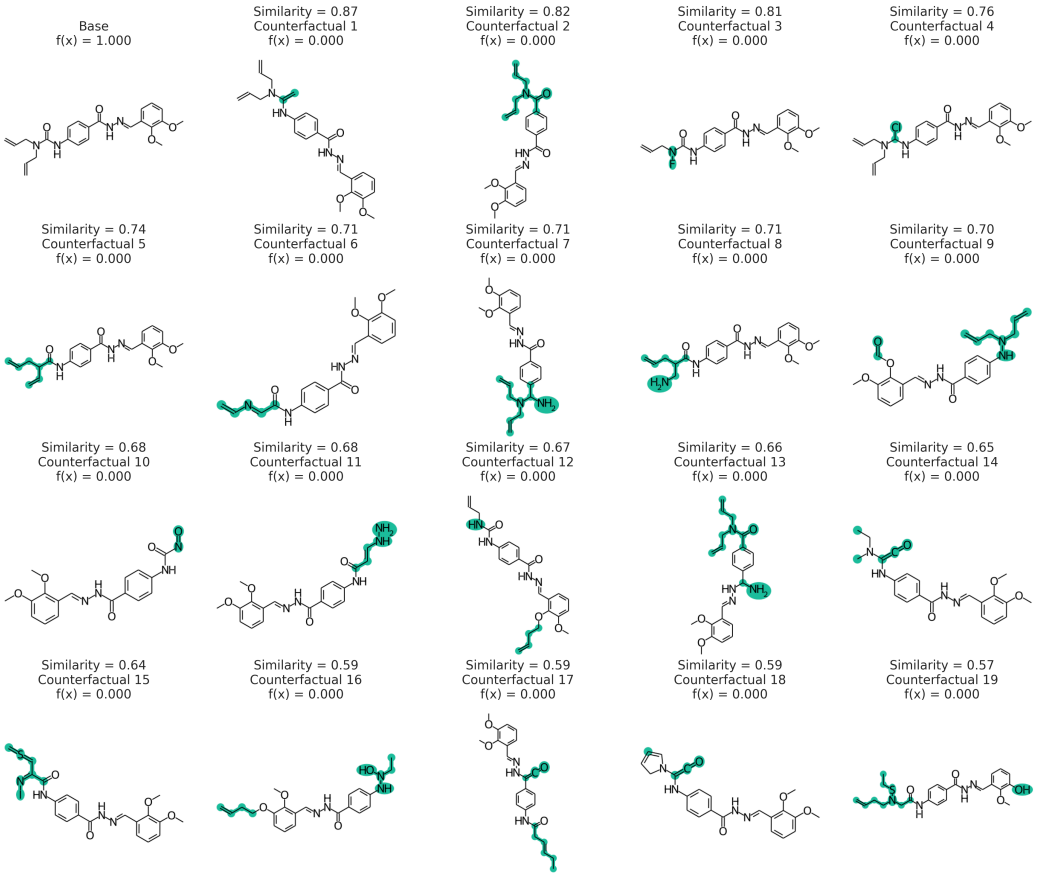

Figure S4: Additional counterfactuals for the GCN model for predicting HIV activity. Top 19 counterfactuals for the base molecule are illustrated here.