CS230

A Deep Learning Approach to Predicting Bioactivity of Small Molecules Based on Molecular Structure

Edward Gao
Stanford University
yxgao19@stanford.edu

Abstract

Computational techniques that can predict molecular properties based on chemical structure has the potential to significantly accelerate drug discovery. Herein, I explore DL approaches to predicting the bioactivity of small molecules from structural data by systematically comparing different molecular representations and model architectures. A word embedding strategy for molecule featurization combined with a 3-layer DNN model proved the most successful. Error analysis shows that improving the dataset labeling scheme may lead to better model performance.

1 Introduction

Drug discovery efforts traditionally involve iterative round of chemical synthesis and biological assaying. This process can be time-consuming, expensive, and oftentimes serendipitous [1]. Computer-aided drug design techniques have shown great promise in alleviating this problem. Machine learning presents an especially attractive technique because properties of a novel small molecule can be extrapolated directly from existing data, obviating expensive physics-based simulations.

In this project, I aim to use a deep learning approach to predict the bioactivity of small molecules based solely on molecular structure. A robust structure-to-activity pipeline can have many important use cases in drug discovery. For example, novel drugs targeting a protein can be quickly developed in response to drug resistance. Off-target effects of a new drug candidate can also be quickly tested by computationally screening against a large panel of common off-target sites.

I will use molecular structures represented as SMILES strings as algorithm inputs (see Dataset section for details). Interestingly, there is no general consensus on the best way to describe molecular structure. Therefore, I will systematically compare a few different ways to encode a molecule and examine how they impact neural network performance. Various model architectures will be applied to molecular feature vectors, and the algorithm will classify each molecule into one of several categories, depending on the biological context of the specific bioactivity value of interest.

As a proof of concept, I will first build a model to predict the inhibitory activity of small molecules against the tyrosine kinase epidermal growth factor receptor (EGFR). Mutations of EGFR can lead to uncontrolled cell proliferation and are found in a number of cancers, and inhibition of EGFR has become one of the most common cancer treatment strategies [2, 3]. This target was chosen because of its clinical relevance and the large amount of data available, although ideally the model developed should be generalizable to any biomolecule target and any bioactivity.

CS230: Deep Learning, Fall 2021, Stanford University, CA. (LateX template borrowed from NIPS 2017.)

2 Related Work

Deep learning techniques have been applied to many different aspects of drug discovery, including target validation, compound property prediction, and retrosynthetic analysis [4, 5]. Previous work on structure-to-activity prediction adopted a number of different methods. Espinoza *et al.* reports that a deep neural network (DNN) architecture that uses 1D feature vectors of molecules as inputs demonstrates promising performance [6]. Each input vector includes information on molecular weight, structure, and physical properties. Joo *et al.* [7] and Ozturk1 *et al.* [8] uses similar input vectors but chooses 1D convolution neural network (CNN) models instead. In contrast, Liu *et al.* uses a 2D CNN architecture instead and encodes each molecular structure as a 2D one-hot feature map [9]. Finally, Sakai *et al.* represents each molecule as a graph to better capture structural information and uses a graph convolutional neural network (GCN) model [10].

3 Dataset

Data used for training and testing were obtained from the ChEMBL dataset [11, 12]. Each molecule is represented as a SMILES string, a system for encoding a molecule as a 1D sequence of letters and symbols. The half maximal inhibitory concentration (IC_{50}) was selected as a measure of compound inhibitory activity against a protein. IC_{50} represents the minimum compound concentration needed to cause a 50% reduction in a protein's activity. Therefore, stronger inhibitors have lower IC_{50} values.

For EGFR, there are 13257 molecules with valid SMILES strings and IC_{50} values. Each molecule was classified as a strong inhibitor, weak inhibitor, or non-inhibitor (Table 1):

Table 1. Summary of EGFR IC_{50} Data

Classification	IC_{50} range	number of examples
strong inhibitor	< 100 nM	5403
weak inhibitor	100 nM - 10 μM	5198
non-inhibitor	$\geq 10\mu$M	2656

Notably, non-inhibitors are underrepresented. This is not surprising because only potent inhibitors tend to get published. To better balance the dataset, more molecules that are not EGFR inhibitors were added. Assuming that a random molecule is unlikely to be an EGFR inhibitor, 839327 drug-like molecules were obtained from ChEMBL, and 2644 of these molecules were randomly selected and classified as non-inhibitors. This augmentation procedure produced a dataset with 15901 examples.

It is also important to note that most proteins do not have thousands of ligands characterized. Any useful machine learning model must function well on much smaller datasets as well. Therefore, I also selected chymotrypsin as a protein example that only has a small number of inhibitors identified (Table 2). No additional data augmentation was performed for this protein. All models will be evaluated on this smaller dataset as well.

Table 2. Summary of Chymotrypsin IC_{50} Data

Classification	IC_{50} range	number of examples
strong inhibitor	< 500 nM	20
weak inhibitor	500 nM - 10 μM	15
non-inhibitor	$\geq 10\mu$M	48

4 Molecular Representations and Learning Methods

Although SMILES strings are lossless representations of molecular structures, they are unstructured and not amenable to many machine learning techniques. Inspired by prior work, I decided to experiment with three different ways to encode SMILES strings. These techniques and their associated learning methods are summarized in Figure 1 and explained more in detail below.

4.1 One-hot SMILES encodings

A naive approach, similar to the method of Liu *et al.* [9], is to simply convert each SMILES string to a one-hot feature map, as only a small number of unique characters are used in SMILES strings. The

generated 2D images can then be easily processed by a 2D CNN. Note that because SMILES strings have varying lengths, it is necessary to pad all strings in a dataset to the same length.

4.2 Molecular fingerprints

Bioinformatics researchers have designed algorithms that can "fingerprint" each molecular structure. Briefly, key chemical features are identified in each structure, and the collection of features present is hashed to generate a 2048-bit binary vector. This strategy is attractive because it produces input vectors of the same length regardless of molecular complexity, and structurally similar molecules are guaranteed to generate similar vectors. For this project, I chose to use the fingerprint function implemented in the RDKit package [13]. The feature vectors obtained were then used as input for a simple logistic regression classifier as well as DNN and 1D CNN models.

4.3 Word embeddings

Borrowing ideas from natural language processing, researchers have developed methods to generate word embeddings for SMILES strings using the Word2Vec technique. In this project, I used the Mol2Vec package [14] and a pre-trained model to embed SMILES strings as 300-dimensional vectors. This approach takes advantage of transfer learning to improve model performance on smaller datasets. The smaller number of features also helps reduce overfitting. Similar to fingerprints, these embeddings can also be input into logistic regression classifiers or DNN and 1D CNN models.

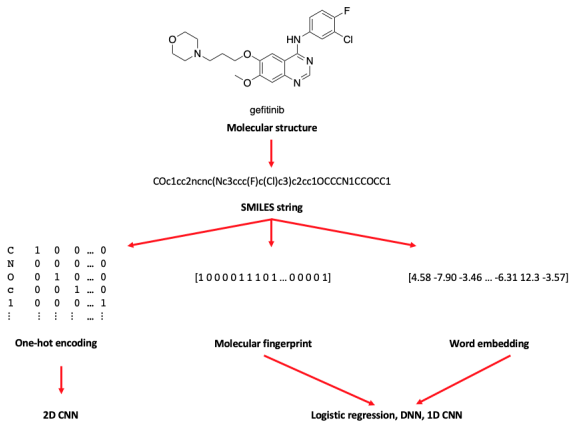

Figure 1. Summary of molecular representation strategies and machine learning architectures explored.

5 Experiments, Results, and Discussion

Logistic regression classifiers were implemented using scikit-learn [15], and all deep learning models were implemented using pytorch [16]. Accuracy was used as the evaluation metric for all models and datasets. All models were optimized using Adams optimization with He initialization and cross entropy loss. To standardize the comparison across different encoding strategies, model architectures, and datasets, similar hyperparameters were used in all cases. For the larger EGFR dataset, 6% of the data was randomly selected each time to serve as the test set. For the smaller chymotrypsin dataset, 20% of the data was used for testing. Because of the small dataset size, the random train/test partition led to high variability in the final accuracies achieved. As such, all models were evaluated on the chymotrypsin dataset 50 times, each time with a different random train/test partition, and the average accuracy was reported. Overall, I found that smaller learning rates led to better performance, as long as the model was trained for enough epochs. Given the size of the dataset, training could still be completed within a reasonable time. In addition, I found that minibatch gradient descent led to significant oscillations in loss. After experimenting with several values, I determined that a batch size of 128 gave a satisfactory balance between performance and training efficiency. Note that for the smaller chymotrypsin dataset, this equates to batch gradient descent.

All training and testing results are summarized in Table 3 and discussed in more detail below.

5.1 One-hot SMILES encodings

To process the one-hot feature maps, I elected to use a 2D CNN model with two convolution layers with ReLU activation and max pooling followed with three fully connected layers. Test accuracy mostly plateaued after 10 epochs of training. Model performance on the larger EGFR dataset was modest at best, and both train and test accuracies on the smaller chymotrypsin dataset was quite poor (Table 3). This result was not entirely surprising. The one-hot 2D input vectors are sparse by definition, and it is reasonable that a CNN architecture cannot learn very well from these data.

5.2 Molecular fingerprints

Using molecular fingerprints as input, a simple logistic regression model achieved quite promising performance. It is evident that the model overfits despite L2 regularization. The variance problem was more prominent for the smaller chymotrypsin dataset: while perfect accuracy was achieved for the training set, the test set only gave 66% accuracy. This can be explained by the large number of input features (2024) relative to the number of examples ($\approx$ 15000 for EGFR, 100 for chymotrypsin).

A DNN model with two hidden layers (of size 500 and 200 respectively) performed marginally better on the EGFR dataset but similarly suffered from high variance on the chymotrypsin dataset. Additional L2 or dropout regularization was unable to alleviate the problem (data not shown).

1D CNNs were used in some previous reports [7, 8]; however, in this case, a model with two 1D convolution layers followed by three fully connected layers gave poorer performance than logistic regression and DNN on both datasets. One potential issue is that the input fingerprint vectors are still sparse. In addition, while CNN models can be powerful in image processing applications where different regions of the input may encode similar information, there is no guarantee that different elements of the molecular fingerprints are significantly related, so the benefit of weight sharing is not evident in this case.

5.3 Word embeddings

The same model architectures were used to process word embeddings. Compared to molecular fingerprints, this approach led to noticeably shorter training times, likely due to the smaller number of input features and the fact that all input features are normalized real values as opposed to binary digits. Interestingly, logistic regression on word embeddings gave slightly worse performance than molecular fingerprints. The DNN model, in contrast, performed better than logistic regression on both datasets and led to a slight improvement over the fingerprint representation. The 1D CNN model again gave poor results. Although the input vectors are no longer sparse in this case, different segments of the input still do not necessarily share significant similarities, making the CNN architecture less useful.

Table 3. Accuracies Achieved Using Different Encodings and Model Architectures

Encoding	Model	EGFR		Chymotrypsin	
		Train	Test	Train	Test
One-hot	2D CNN	0.756	0.687	0.497	0.505
Fingerprint	Logistic Regression	0.871	0.733	1.0	0.656
	DNN	0.881	0.775	0.885	0.529
	1D CNN	0.721	0.676	0.518	0.489
Word embedding	Logistic Regression	0.721	0.676	0.985	0.616
	DNN	0.888	0.769	0.923	0.666
	1D CNN	0.665	0.646	0.448	0.428

5.4 A closer look at word embeddings

In summary, logistic regression on molecular fingerprints and DNN on word embeddings performed the best. The word embedding approach was especially interesting because of its faster training times and better performance on smaller datasets, so I decided to examine its performance more closely.

This model clearly overfit the training data, as evidenced in the discrepancy between the training and test accuracies. Unfortunately, regularization techniques such as L2 regularization and dropout were unable to improve test set performance. Plotting training and test accuracies against training epoch shows a more nuanced story (Figure 2). The training accuracy continued to increase as training

progressed, as is typical for an overfitting model. The test accuracy, however, quickly plateaued and did not decrease. This suggests that the underlying issue is more complicated than simply overfitting. Perhaps there are limitations in the model architecture or the input data that lead to poor performance.

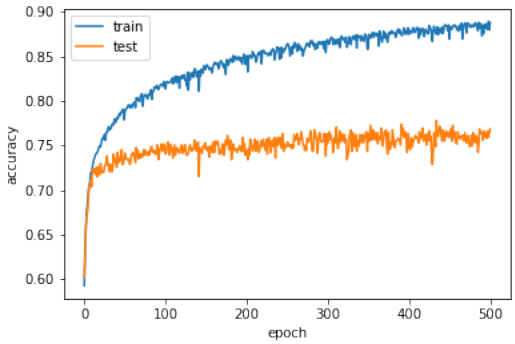

Figure 2. Training and test accuracies as a funciton of training epoch for the DNN model on word embeddings of the EGFR dataset.

An examination of the confusion matrix (Table 4) provides one possible explanation. It is clear that the model is quite good at distinguishing between strong EGFR inhibitors from non-inhibitors. The majority of the error comes from the intermediate category. This makes sense as the classification scheme used for this dataset is quite arbitrary. For example, two structurally similar molecules may have IC_{50} values of 99 and 101 nM, respectively, but they would have been assigned different labels in this dataset. It is rather unreasonable to expect the algorithm to label them differently. Ultimately, any classification scheme must draw the line somewhere, and this problem would persist.

Table 4. Confusion Matrix of Predictions on the EGFR Dataset

		predicted		
		strong inhibitor	**weak inhibitor**	**non-inhibitor**
actual	**strong inhibitor**	260	65	2
	weak inhibitor	64	201	34
	non-inhibitor	11	45	273

6 Conclusion and Future Work

In this project, I explored using deep learning to predict the bioactivities of small molecules directly from chemical structure. After comparing several molecular representations and machine learning models, I concluded that a molecular word embedding strategy combined with a simple 3-layer DNN model gave the best performance. This approach takes advantage of transfer learning from a pre-trained model and offered faster training times and the most consistent performance across larger and smaller datasets. Error analysis showed that a key limiting factor in model performance is the labeling scheme of the input data. Future work should aim to devise a better labeling strategy, perhaps by curating a dataset with more clearly demarcated categories. For example, a training set containing only the strongest inhibitors and non-inhibitors may lead to better performance. Furthermore, additional strategies for encoding molecular structures should be explored as well.

7 Contributions

I worked on this project alone.

Code Availability

All source code can be found at `https://github.com/yixuan-edward-gao/cs230_project`.

References

[1] Pammolli, F., Magazzini, L., & Riccaboni, M. (2011) The productivity crisis in pharmaceutical R&D. *Nature Reviews Drug Discovery* **10**(6):428-438.

[2] Zhang, H., Berezov, A., Wang, Q., Zhang, G., Drebin, J., Murali, R., & Greene, M. I. (2007) ErbB receptors: from oncogenes to targeted cancer therapies. *The Journal of Clinical Investigation* **117**(8):2051-2058.

[3] Sigismund, S., Avanzato, D., & Lanzetti, L. (2018) TEmerging functions of the EGFR in cancer. *Molecular Oncology* **12**(1):3-20.

[4] Vamathevan, J. *et al.* (2019) Applications of machine learning in drug discovery and development. *Nature Review Drug Discovery* **18**(6):463-477.

[5] Chen, H., Engkvist, O., Wang, Y., Olivecrona, M., & Blaschke, T. (2018) The rise of deep learning in drug discovery. *Drug Discovery Today* **23**(6):1241-1250.

[6] Espinoza, G. Z., Angelo, R. M., Oliveira, P. R., & Honorio, K. M. (2021) Evaluating Deep Learning models for predicting ALK-5 inhibition. *PLos ONE* **16**(1):e0246126.

[7] Joo, M. *et al.* (2019) A Deep Learning Model for Cell Growth Inhibition IC50 Prediction and Its Application for Gastric Cancer Patients. *International Journal of Molecular Sciences* **20**(24):6276.

[8] Ozturk1, K., Ozgur1, A., & Ozkirimli, E. (2018) DeepDTA: deep drug–target binding affinity prediction. *Bioinformatics* **34**:i821–i829.

[9] Liu, P., Li, H., Li, S., & Leung, K.-S. (2019) Improving prediction of phenotypic drug response on cancer cell lines using deep convolutional network. *BMC Bioinformatics* **20**:408.

[10] Sakai, M., Nagayasu, K., Shibui, N., Andoh, C., Takayama, K., Shirakawa, H., & Kaneko, S. (2021) Prediction of pharmacological activities from chemical structures with graph convolutional neural networks. *Scientific Reports* **11**:525.

[11] Gaulton, A. *et al.* (2017) The ChEMBL database in 2017. *Nucleic Acids Research* **45**(D1):D945-D954.

[12] Davies, M. *et al.* (2015) ChEMBL web services: streamlining access to drug discovery data and utilities. *Nucleic Acids Research* **43**(W1):W612-W620.

[13] RDKit: Open-source cheminformatics; http://www.rdkit.org.

[14] Jaeger, S., Fulle, S., & Turk, S. (2018) Mol2vec: Unsupervised Machine Learning Approach with Chemical Intuition. *Journal of Chemical Information and Modeling* **58**(1):27-35.

[15] Pedregosa, F. *et al.* (2011) Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* **12**:2825-2830.

[16] Paszke, A. *et al.* (2019). PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Advances in Neural Information Processing Systems 32*: 8024–8035. Curran Associates, Inc.