Deep Learning for Network Chemistry and Network Biology

A Project Report

submitted by

SACHIN AGRAWAL

in partial fulfilment of the requirements
for the award of the degree of

BACHELOR OF TECHNOLOGY &
MASTER OF TECHNOLOGY

DEPARTMENT OF ELECTRICAL ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY, MADRAS.
May 2019

THESIS CERTIFICATE

This is to certify that the thesis entitled **Deep Learning for Network Chemistry and Network Biology**, submitted by **Sachin Agrawal** (**EE14B104**), to the Indian Institute of Technology, Madras, for the award of the degree of **Bachelors of Technology** and **Master of Technology**, is a bona fide record of the research work carried out by him under my supervision. The contents of this thesis, in full or in parts, have not been submitted to any other Institute or University for the award of any degree or diploma.

Dr. Balaraman Ravindran
Research Guide
Professor
Dept. of Computer Science and Engineering
IIT-Madras, 600 036

Dr. Venkatesh Ramaiyan
Research Guide
Assistant Professor
Dept. of Electrical Engineering
IIT-Madras, 600 036

Dr. Karthik Raman
Research Guide
Associate Professor
Dept. of Biotechnology
IIT-Madras, 600 036

ACKNOWLEDGEMENTS

I am grateful to my primary thesis advisor Dr. Balaraman Ravindran from the Department of Computer Science and Engineering at IIT Madras. Over the past year, I have had the opportunity to explore a plethora of topics thanks to him. I wish to thank him for his constant support and guidance regarding my thesis. I also wish to thank him for many meaningful discussions on the latest developments in deep learning and where AI research is heading.

I must also thank Prof. Karthik Raman from the Department of Biotechnology, for his useful and insightful inputs on the Chemical Reaction Prediction problem, which has formed a crucial part of my thesis.

I would also like to thank my co-guide and faculty advisor, Prof. Venkatesh Ramaiyan from the Department of Electrical Engineering, for advising me over the past five years and for his guidance during my institute life.

I also wish to thank Priyesh Vijayan for mentoring me and showing me how to do meaningful research. On days when I reached a dead-end in my research, a discussion with Priyesh would open up new avenues. Priyesh has also been a constant source of inspiration for me by demonstrating that it is possible to work on an infinite number of topics simultaneously.

I must also acknowledge Aakash Srinivasan for being my partner in crime in our collaboration with Baker Lab. It has been a pleasure discussing and debating ideas with you.

I also wish to thank my collaborators at Baker Lab at University of Washington, namely, Prof David Baker and Qian Cong for giving me the opportunity to work on the Protein-Protein interaction prediction problem.

I also wish to thank the networks group, namely, Tarun Kumar, Nikita Moghe, Sankaran Vaidyanathan and Deepak Maurya, for exposing me to the various problems which are being solved using the combination of networks and deep learning. It has been a pleasure learning about your research.

Lastly, but definitely not the least, I would like to thank my parents and my sister for consistently supporting and guiding me through my life and studies.

ABSTRACT

KEYWORDS: Deep Learning, Networks, Chemical Reaction Prediction, Protein-Protein Interaction Prediction, Graph Convolutional Networks, Drug Design

In many real life systems, we find entities are influenced by each other, rather than being independent. Such systems can be effectively modeled as graphs, with the entities forming nodes, and the interaction being represented with edges. In recent times, there has been a surge in research applying learning algorithms to such network data. We find many chemical and biological data which can be effectively modeled using graphs. We review the applications of deep learning on such data. We discuss the benefits of such applications in optimising the drug design pipeline.

We also demonstrate state of the art results for the problem of chemical reaction prediction using an optimised graph convolutional network architecture. Our model beats the baseline, while having 25x lesser learning parameters.

We also propose a novel setup for protein-protein interaction prediction which is able to effectively combine information from multiple noisy incomplete graphs with node attribute information. We believe that our setup will allow us to predict co-functioning as well as co-evolving pairs of proteins.

We hope that this report informs the reader regarding the various applications of deep learning to network biology and network chemistry. Further, we hope

that our work in the area of reaction prediction and protein interaction prediction inspires researchers to work on these problems.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABBREVIATIONS

GCN	Graph Convolution Network
GNN	Graph Neural Network
QSAR	Quantitative Structure Activity Relationship
PPI	Protein-Protein interaction
DDI	Domain-Domain interaction
DNA	Deoxyribonucleic Acid
RNA	Ribonucleic Acid
mRNA	messenger Ribonucleic Acid
RNN	Recurrent Neural Network
GRU	Gated Recurrent Unit
VAE	Variational Auto Encoder
GAN	General Adversarial Network
SMILES	Simplified molecular input line entry specification
LEF	Linear electron flow
CN	Common Neighbors
AA	Adamic Adar
JA	Jaccard Index
MI	Mutual Information
RA	Resource Allocation
PA	Preferential Attachment
PF	Propflow

WYS	Watch Your Step
GO	Gene Ontology
MI	Mutual Information
AUROC	Area Under Receiver Operating Characteristics
AUPR	Area Under Precision-Recall
GREMLIN	Generative REgularized ModeLs of proteINs
SVD	Singular Value Decomposition
ROC	Receiver Operating Characteristics
PR	Precision-Recall
USPTO	United States Patent and Trademark Office
DTA	Drug-Target Affinity
LSTM	Long Short Term Memory
RL	Reinforcement Learning
KEGG	Kyoto Encyclopedia of Genes and Genomes
DNN	Deep Neural Network
ANN	Artificial Neural Network

NOTATION

G	Graph
V	Set of Vertices. The number of vertices is given by $\|V\|$ or n.
E	Edge set. The number of edges is given by $\|E\|$ or m.
A	Adjacency Matrix. $A \in \mathbb{R}^{n \times n}$.
D	Degree Matrix. It is diagonal matrix with degree of node as diagonal entries.
I	Identity Matrix.
$c_i^{(u)}$	Feature vector for atom u after i layers of GCN.
c_{uv}	Feature vector for bond between atoms u and v.
$N(v)$	Set of neighbors of node v.
τ	A non-linear function, typically a fully-connected neural network
S	State of a system.
A	Action taken by an RL agent.
P	Transition function dependent on previous state and action.
R	Reward function.
T	Training set of molecules.
V	Set of valid molecules generated.
V_u	Set of valid and unique molecules generated.

CHAPTER 1

INTRODUCTION

We have seen the tranformational impact of deep learning in many domains. In domains such as computer vision and speech recognition, deep learning has become so ubiquitous, that majority of researchers have shifted their primary focus from conventional methods to deep learning methods [Voulodimos *et al.*, 2018]. Deep learning has been applied to different types of data. Most commonly, deep learning has been applied to images, time series data and text data. More recently, there has been an increasing interest in the application of deep learning to data which can be represented as graphs. Graphs can be used to represent molecules, protein interaction networks, social interactions and many other common systems. In this thesis, we will focus on the application of methods which lie at the intersection of deep learning and networks to the fields of biology and chemistry. Further, we will provide state of the art results in the treatment of two problems, namely, chemical reaction prediction and protein-protein interaction prediction.

1.1 Deep Learning on Networks

In the last few years, many papers have visited the problem of applying neural networks to arbitrary networks [Kipf and Welling, 2016; Grover and Leskovec, 2016; Abu-El-Haija *et al.*, 2017; Vijayan *et al.*, 2018]. The goal of deep learning on networks can be node classification, link prediction, community detection, clustering of nodes or graph classification.

Graph data consists of nodes and links. Nodes and links may or may not have attributes. For example, in the case of molecules, both nodes (atoms) and links (bonds) have attributes. In case of PPI networks, only nodes have attributes. Applying deep learning to graph data is difficult, because we need to learn the network information along with the attribute information. Furthermore, in some networks the global properties of the network are more important, while in others the local network properties are more important. The challenge is to learn an appropriate feature vector representation for each node, that encodes both, the attribute information and relevant network information.

Traditional machine learning algorithms on networks required hand-crafted features. These hand-crafted features could be summary node statistics such as degree and clustering coefficients [Henderson *et al.*, 2012; Bhagat *et al.*, 2011] or graph kernels [Vishwanathan *et al.*, 2010]. The problem with these approaches is the same problem as with typical machine learning approaches: they require one to design these features for each problem. If one set of features give abysmal results for a particular problem, then one must understand why given set of features fail and come up with new features. This whole process can be taxing and time consuming. Therefore, we require a solution which can automate the feature engineering process.

Scarselli *et al.* [2009] introduced graph neural networks, which allow propagation of information between adjacent nodes. However, this model was limited to graphs which can be fitted into memory. This becomes a problem for larger networks. Graph Convolution Networks (GCNs) [Kipf and Welling, 2016] recursively convolves one-hop neighborhood information with a symmetric graph Laplacian allowing them to handle larger networks as well. Vijayan *et al.* [2018]

extended GCN architecture to regulate attribute and network information from different distances independently. Schlichtkrull *et al.* [2018] extended GCNs for multi-relational graphs,i.e for graphs with multiple types of edges. Veličković *et al.* [2018] add self-attention [Bahdanau *et al.*, 2015] to GCNs for node classification problems.

Many recent state of the art papers on network data make use of GCNs or some variation of GCNs for various tasks. Primarily, GCNs are used to learn an efficient representation of nodes. This node representation is then used to learn the task at hand which maybe node classification, link prediction, community detection, clustering of nodes or graph classification.

1.2 Networks in Chemistry

Chemistry is the branch of science concerned with the substances of which matter is composed, the investigation of their properties and reactions, and the use of such reactions to form new substances. Essentially, chemistry is the study of molecules, molecular properties and reactions between molecules.

In 2012, a team of deep learning researchers won the Merck Molecular Activity Challenge [Dahl *et al.*, 2014], without having a single chemist or biologist on their team. Since then, there has been an exponential rise in the application of deep learning to chemistry [Goh *et al.*, 2017*a*].

1.2.1 Molecular Networks

Any molecule can be represented as a graph, with atoms as nodes and bonds as edges. Molecules can also be represented as strings using the SMILE representation of molecules. However, this representation is not robust. A single character perturbation in the SMILE representation can lead to a significant change in the graph [You *et al.*, 2018*a*]. Hence, a graph based representation of molecules is preferred for computational methods. In general there are three applications of analysing molecular networks, namely, reaction prediction, Quantitative structure activity relation (QSAR) and generation of molecules with desired properties.

Reaction Prediction

Efficient and accurate chemical reaction prediction is essential for any drug design pipeline [Engkvist *et al.*, 2018]. Chemical reaction prediction is also useful for discovering cheaper pathways to any molecule. Computational methods for chemical reaction prediction can help reduce the cost of manufacturing various chemicals. The reaction prediction problem is two-fold. The first problem is to predict the product of a reaction, given the reactant molecules. The second problem is to find reactant molecules, given a product molecule. This second problem is referred to as the retro-synthesis problem. Both problems can be attempted by applying deep learning methods to molecular networks.

Quantitative Structure Activity Relationship

Quantitave Structure Activity Relationship (QSAR) models are regression or classification models to predict certain properties of a molecule. Common properties of interest are toxicity, solubility, boiling points, drug-likeness and so on. Machine learning approaches to QSAR require feature engineering based on the molecule structure. Deep learning papers for QSAR models generally apply a convolutional network on the molecular graph, and learn a classifier or regressor on top of the node embeddings learned [Li *et al.*, 2018; Wu *et al.*, 2017].

Molecule Generation

One of the many challenges in drug design is the sheer size of the search space for novel molecules. It has been estimated that 10^{60} drug-like molecules could possibly be synthetically accessible [Ruddigkeit *et al.*, 2012]. The process of drug design relies heavily on existing datasets of drug-like molecules, which form a very small subset of the large molecular space.

Generative models can help us traverse the space of novel molecules not available in existing databases. The process of molecule generation can be goal-directed or un-directed. In un-directed molecule generation, the model is just trying to learn the space in which drug-like molecules lie. We can then sample from this space to generate new molecules. These molecules are unlikely to be useful as they do not have the properties desired by us. Goal-directed molecule generation generates molecules with specific desirable properties.

In recent times, researchers have applied GRUs [Chung *et al.*, 2014], VAEs [Kingma and Welling, 2014] and GANs [Goodfellow *et al.*, 2014] for molecule generation. In Chapter 2, we cover each of these approaches in more detail.

1.2.2 Reaction Networks as Hypergraphs

Another way to model reactions using networks, is by denoting molecules as nodes, and reactions as directed hyperedges [Fagerberg *et al.*, 2018]. In this approach the reactants and product form a directed hyperedge, with the reactants on one side, and the products on the other side of the directed hyperedge.

This method of modelling is generally used when the goal is to search for synthesis pathway for a given molecule. It can also be used to search for specific chemical reactions, which can be useful for reaction prediction for some systems. However, such a system will fail if the input is not present in the dataset. This method of modelling can also be used to pose the problem of reaction prediction as a link prediction problem [Yadati *et al.*, 2019]. Such a system is very useful for modelling systems such as metabolic networks, where all nodes are known beforehand.

1.3 Networks in Biology

Network biology involves the study of the complex interactions of bio-molecules that contribute to the structures and functions of living cells [Camacho *et al.*, 2018]. In recent times, there has been an increase in the amount and complexity of data collected in biology and health-care. We require efficient algorithms to analyze this data and extract useful insights from it.

The most common types of networks in biology are:

1.3.1 Protein-Protein Interaction Networks

Protein-protein interactions are physical contacts between two or more proteins. Proteins allow organisms to function. However, proteins do not function independently of each other. Proteins interact with genes and other proteins in order to perform their functions. PPI networks are used to represent interactions among proteins in a cell or a tissue. PPI networks have been used to identify novel protein functions [Peng *et al.*, 2014]. Further, PPI networks can be used to identify mechanisms for various processes which can help identify progression of various diseases [Fionda, 2018].

Typically protein interactions are represented using dyadic connections. However, very often protein complexes comprise of more than two proteins. In such cases a hypergraph representation of protein complexes maybe more accurate [Ramadan *et al.*, 2004].

Another network, very similar to PPI network is the Domain-Domain interaction (DDI) network. Domains are building blocks of proteins. They are blocks that fold independently. A DDI network represents exactly which domains between two proteins are interacting with each other. DDI networks are much larger compared to PPI networks as they contain more number of nodes [Kim *et al.*, 2012].

1.3.2 Metabolic Networks

Metabolic networks are used to represent the complete set of metabolic and physiological processes that occur in a cell. Some common data sources for metabolic networks are KEGG [Morishima *et al.*, 2018], Ecocyc [Kothari *et al.*, 2016] and BioCyc [Karp *et al.*, 2017]. Analysis of metabolic networks can help us find new metabolic pathways. This helps us improve our understanding of biological functioning.

One of the ways to model metabolic networks is using hypergraphs as in Mithani *et al.* [2009]. In the hypergraph representation, nodes represent substances (molecules), and hyperedges represent reactions. Another method of modelling metabolic networks is by using molecular networks as in Sankar *et al.* [2017]. In this method the molecules in the reaction are treated as graphs. In either case, the analysis of metabolic networks allows us to find new metabolic pathways. Another goal metabolic network analysis can be community detection. Generally metabolic networks are highly modular, and the various communities can be indicative of the functional significance of the reactions as shown in Guimerà and Amaral [2005].

1.3.3 Gene Regulatory Networks

Some types of proteins interact with mRNA. These types of proteins are referred to as transcription factors. This interaction between mRNA and protein is essential for gene expression in organisms. Gene regulatory networks allow us to capture this information. A gene regulatory network (GRN) is a collection of molecular regulators that interact with each other and with other substances in the cell to govern the gene expression levels of mRNA and proteins. The regulator can be DNA, RNA, protein and complexes of these.

Gene regulatory networks can be represented using boolean networks [Martino *et al.*, 2007], i.e the edges are un-weighted in the graph. However, this causes loss of information since the experiments used to form the network return probabilistic values. To overcome this some papers use Bayesian networks as in Friedman *et al.* [2000].

1.3.4 Cell Signaling Networks

Cells are generally specialised to perform a specific function. In order for an organism to perform biological processes, cells must work together. In order to collaborate cells must be able to signal each other. This signalling generally takes place with the help of proteins.

Cell signaling networks are modelled using molecules as nodes and interactions as edges. These molecules are generally proteins, but can be lipids, substrates or metabolites. The interactions between two molecules may lead to activation or deactivation of certain substances Fionda [2018].

Generally cell signaling network to be spatially uniform and model the time variation of concentrations with the help of ordinary differential equations Szabo *et al.* [2011]. Some models divide the cell into compartments as in Păun [2000] .

1.3.5 Drug Target Interaction Networks

A drug is a substance which when taken in (intake maybe inhalation, oral, absorption by patch or injection) by a person, alters the functioning of the body. Drugs generally bind to a specific protein site. The use of computational methods to identify novel drug target interactions is of great interest. This can help us cure certain types of diseases and design better drugs for existing diseases.

Conventional methods for drug design are time taking and expensive. Recently, there has been an interest in applying deep learning for designing drugs as in [ztrk *et al.*, 2018; Vilar *et al.*, 2016; Ragoza *et al.*, 2017]. These methods model proteins and drugs as molecular networks and try to determine binding affinity between the two. These methods will be covered in more detail in chapter 4 of this thesis.

In this thesis, I will focus on research which applies deep learning methods to network biology. Primary focus will be on PPI networks, metabolic networks and drug target interaction networks. There is a lack of papers which apply deep learning models to other types of biological networks. This could be a fruitful direction for future research. I will also be providing novel results and analysis for the *Escherichia coli* protein-protein interaction network.

1.4 Contributions of this Thesis

The contributions of this thesis are three-fold:

1. We provide a review of the current methods and applications of deep learning to network biology and network chemistry. With an increasing amount of work being done in the field, this review summarises previous works done in the field. This thesis does not provide a comprehensive review of deep learning architectures on networks. The reader is referred to [Hamilton *et al.*, 2017*b*; Yang *et al.*, 2015; Wu *et al.*, 2019*b*] for the same.

2. We provide state of the art results for the chemical reaction prediction problem. We attempted the chemical reaction prediction for Lowe's USPTO reaction database [Lowe, 2012].

3. We provide a new setup and state of the art results for the PPI prediction problem. We provide results for the PPI network corresponding to *Escherichia coli* organism.

1.5 Outline of this Thesis

The remainder of this thesis is structured a follows. In Chapter 2, we provide a survey of literature which applies deep learning to network chemistry. Focus will be on three problems, namely, reaction prediction, QSAR and molecule generation.

In Chapter 3, we analyse the chemical reaction prediction problem in more detail. We give special attention to Jin *et al.* [2017*b*], and improve on their result by providing a better neural architecture. In chapter 4, we provide a survey of the papers which apply deep learning on biological networks. Special focus will be on PPI networks. In Chapter 5, we provide a novel setup for PPI link prediction, by making use of multi-graphs. In the final chapter, we provide a brief discussion of future works in this area and the potential impact of research in this field.

CHAPTER 2

Survey of Deep Learning Applications in Network Chemistry

In the previous chapter we had a brief look at the methods and applications of deep learning in network chemistry. In this chapter, we will provide a comprehensive survey of the same. This chapter also sets the stage for the next chapter, where we will provide state of the art results for chemical reaction prediction.

Before jumping into the applications of deep learning to network chemistry, we would like to make a comment on the graph representation of molecules.

Network Representation of Molecules: While the network representation of molecules is a powerful abstraction, it still loses some information. Molecules are 3-dimensional in nature. However, graphs only capture a 2-dimensional projection of molecules. Further, information regarding bond strength and electron clouds is also lost. Such information maybe crucial in some cases and it is important to capture this information to build more general models. The exploration of methods which capture the 3-D structure of molecules is an important direction for future work.

2.1 Chemical Reaction Prediction

Over the past 40 years many tools have been developed to aid in material design and synthetic planning. The first computational tool for the task of retro-synthetic planning was LHASA [Corey, 1971]. The software used a large number of subroutines and encoded rules. It took a decade to build. In the 1980s and 90s, various new softwares like SOPHIA [Satoh and Funatsu, 1996], CAMEO [Jorgensen *et al.*, 1990] and EROS [Gasteiger *et al.*, 1987] were developed. Though each of these had their differences, they all functioned based on hand coded rules for chemical reactions. They would identify important functional groups in the molecular graph, and try to match this to existing reaction templates. The first drawback of such models is that their accuracy depends largely on the encoded knowledge base. The second drawback is that we cannot learn anything new from such models.

Another approach for chemical reaction prediction is the use of physical chemistry principles. Softwares such as ROBIA [Socorro *et al.*, 2005] use quantum mechanical calculations for reaction prediction. While such approaches have a wider reach than template based approaches, they are expensive as they require a new set of computations for each reaction family. Hence, they are limited in the size of the dataset and reaction types they can handle.

A third approach for reaction prediction requires predicting the reaction mechanism. These approaches model chemical reactions as interactions between electron donors and electron acceptors. The works which follow this approach are [Kayala and Baldi, 2011; Fooshee *et al.*, 2018]. Both these papers predict the reaction product with the help of two sequential models. The first model identifies potential electron sources and sinks. The second model ranks all combinations of sources and sinks. This approach is template-free. However, it can predict only one step at a time, and needs to be requires multiple iterations for multi-step reactions. Further, it requires a dataset which contains information regarding mechanism of the reaction.

Another template-based approach to reaction prediction makes use of machine learning. In this approach, molecules are represented with the help of molecular descriptors such as morgan fingerprints [Morgan, 1965]. These fingerprints are used to classify the type of reaction. Then, based on the reaction type, a transformation is applied to the reactants to obtain the product [Wei *et al.*, 2016; Gelernter *et al.*, 1990]. Such an approach would however fail for complex reactions which cannot be clearly classified. It would also fail for certain reactants where there maybe multiple reaction centres for the same type of reaction.

All of the papers mentioned above operate on different datasets, and it is difficult to compare the various approaches. In recent times, most papers on reaction prediction make use of the USPTO dataset [Lowe, 2017]. The dataset consists of reactions mined from US patents filed between 1976 and 2016. After removing duplicates and erroneous reactions, the dataset consists of 400K reactions. One drawback of this dataset is that it does not contain the physical conditions in which the reaction takes place. Another point to note, is that the dataset does not denote which molecules take part in the reaction, and which molecules act as reagents or catalysts for the reaction. A detailed analysis of this dataset will be presented in the next chapter.

[Jin *et al.*, 2017*a*; Schwaller *et al.*, 2018; Bradshaw *et al.*, 2019; Do *et al.*, 2019; Coley *et al.*, 2017] all make use of the USPTO dataset for the forward reaction prediction problem. We will cover each of these approaches in some detail here.

Paper Summary: Prediction of Organic Reaction Outcomes Using Machine Learning

Coley *et al.* [2017] uses a template based machine learning approach on the USPTO dataset. The authors automatically extract reaction templates from the given dataset. They then train a neural network to rank the various reaction templates for a given reactant. This approach differs from previous template based approaches like Wei *et al.* [2016] as it automatically extracts templates from reactions. Further, for the purpose of ranking, rather than using molecular descriptors, this paper uses an edit based model, i.e they take the difference between the candidate product and the reactant. Another ranking model which the paper proposes, makes use of just the candidate product (disregarding the reactants), ranking it based on its stability. The final ranking model the paper proposes, ensembles the two approaches.

Paper Summary: Found in Translation: predicting outcomes of complex organic chemistry reactions using neural sequence-to-sequence models

Schwaller *et al.* [2018] uses the SMILE representation of molecules for reaction prediction. The idea is to translate from the reactant SMILE to the product SMILE using a state-of-the-art translation model. This is similar to the work of Nam and Kim [2016], which directly uses the Tensorflow translate model [Abadi *et al.*, 2015] to predict chemical reactions. However, [Nam and Kim, 2016] restrict themselves to textbook reactions, and can be considered to be a toy dataset in comparison to the USPTO dataset. Schwaller *et al.* [2018] uses the neural machine translation algorithm taken from Luong *et al.* [2017] for reaction prediction, and uses the much larger and noisier USPTO dataset. However, they make use of SMILE representations, which can be brittle.

Paper Summary: Predicting Organic Reaction Outcomes with Weisfeiler-Lehman Network

Jin *et al.* [2017*a*] proposes two-step template free model for reaction prediction. The first step uses a graph convolutional network to aggregate node embeddings. These embeddings are used to compute atom pairs which are likely to react with each other. The second part of the model runs a combinatorial search over the candidate pairs to identify likely products and ranks them. Let G be the graph corresponding to the reactant molecule. The initial node embedding ($c_u^{(0)}$) for an atom (u) is a concatenation of features such as atomic number, valency, charge and aromaticity. The edge embedding (c_{uv}) for a bond between a pair of atoms (u and v) is a one hot encoding of the type of bond. To aggregate the neighbourhood information, the model uses a GCN:

$$c_v^{(i+1)} = (U_1 c_v^i + U_2 \sum_{u \in N(v)} \tau(V[c_u^i, c_{uv}])) \tag{2.1}$$

where τ can be any non-linear function. The GCN helps aggregate neighbourhood information. The final representation of atoms comes from mimicking the set matching function in the WL isomorphism test [Douglas, 2011]. The resulting equation is :

$$c_v^N = \sum_{u \in N(v)} W^{(0)} c_u^{(N-1)} . W^{(1)} c_{uv} . W^{(2)} c_v^{(N-1)} \tag{2.2}$$

To aggregate information from global nodes the model computes the attention between each pair of atoms according to

$$\alpha_{uv} = \sigma(u^T \tau(M_a c_u^N + M_a c_v^N + M_b b_{uv})) \tag{2.3}$$

where α_{uv} represents the attention between atoms u and v, and N is the number of layers of GCN applied. Global node embeddings are computed according to

$$\tilde{c_u} = \sum_v \alpha_{uv} c_v^N \tag{2.4}$$

Using global node embeddings, the likelihood of reaction between any pair of atoms can be computed according to

$$s_{uv} = \sigma(u^T \tau(P_a \tilde{c_u} + P_a \tilde{c_v} + P_b b_{uv})) \tag{2.5}$$

where s_{uv} is the probability of reaction between u and v. After identifying the top K atom pairs and corresponding bond orders, the paper runs a combinatorial search for the best combination of edits. A graph edit can be represented as a triple: (u, v, b), where u and v represent atoms and b represents the new bond order between u and v. In the USPTO dataset, the maximum number of edits in a reaction is six. Hence, the combinatorial search is restricted to this size. Each combination of edits less than 6, from the top K edits forms a candidate. To score the candidate, another network is trained on the difference graph of the candidate product and the reactant. The difference graph for product p_i is computed according to

$$d_u^{(p_i,0)} = c_u^{(p_i,0)} - c_u^{(r,0)} \tag{2.6}$$

After applying L layers of GCN according to (2.1), we obtain the node embedding for the difference graph. This is used to score the candidate product as per

$$s(p_i) = sigmoid(M \sum_{v \in p_i} d_v^{(L,p_i)}) \tag{2.7}$$

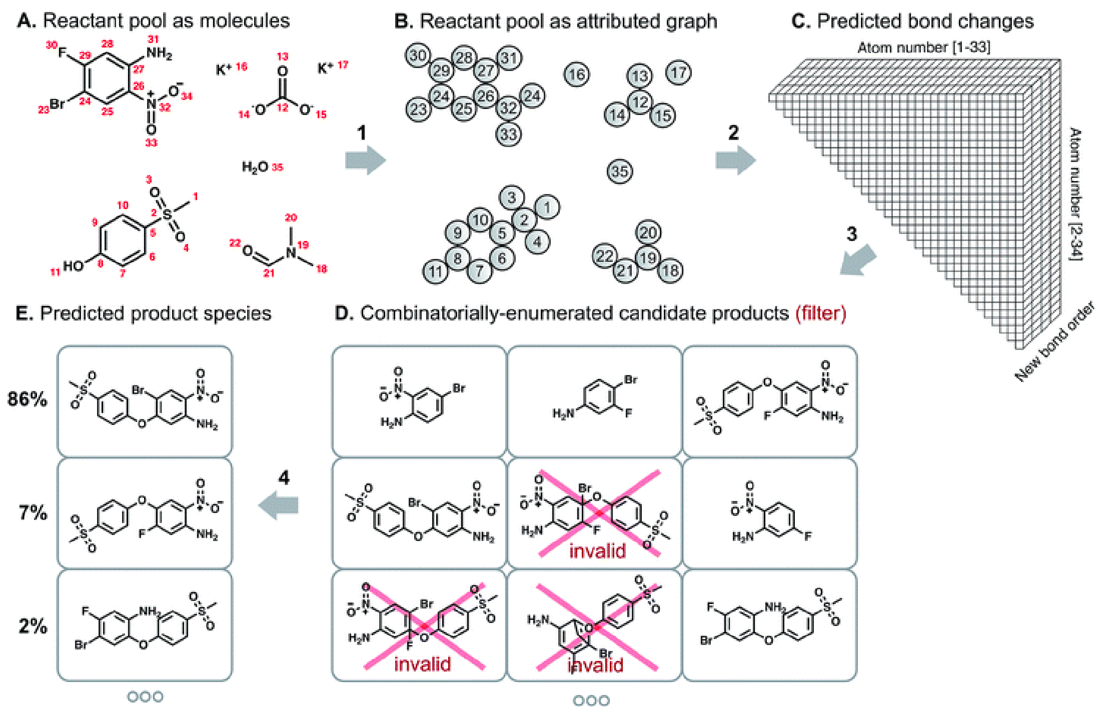
A. Reactant pool as molecules
B. Reactant pool as attributed graph
C. Predicted bond changes
Atom number [1-33]
Atom number [2-34]
New bond order
E. Predicted product species
D. Combinatorially-enumerated candidate products (filter)
86%
7%
2%
invalid
invalid
invalid
1
2
3
4

Figure 2.1: A pictorial summary of Jin *et al.* [2017*a*]

The entire model is summarised in Figure 3.1.

A modification of this paper is presented in Coley *et al.* [2019], where the reactivity score for a given edit obtained from the first model is added to the candidate score ($s(p_i)$). The main drawback of this model is that it cannot be trained end-to-end. The second issue is that the combinatorial search over candidate pairs is expensive. Despite its drawbacks, it is the current state of the art on the reaction prediction problem.

Paper Summary: Graph Transformation Policy Network

Do *et al.* [2019] provides an alternative to the combinatorial search required in Jin *et al.* [2017*a*]. Further, Do *et al.* [2019] is an end-to-end approach for reaction prediction. In Do *et al.* [2019], the authors identify the pairs of atoms which are likely to react, as mentioned in Jin *et al.* [2017*a*]. To identify candidate products, the authors propose a reinforcement learning based approach. Instead of running a combinatorial search, the authors apply the edits in a sequential manner.

The series of graph edits can be represented as a sequence:
$(\xi, u, v, b)^0, (\xi, u, v, b)^1, (\xi, u, v, b)^2, \ldots (\xi, u, v, b)^{(T-1)}, (\xi, u, v, b)^T$
In the above sequence, ξ is a binary signal indicating end of reaction, and T represents the maximum number of transformations allowed. At every step τ, the change $(u, v, b)^\tau$ is applied, provided $\xi^\tau = 1$. This transformation allows us to go from G^τ to $G^{\tau+1}$. This transformation can be viewed as an MDP, characterized by (S, A, P, R, γ), where S is a set of states, A is a set of actions(graph edits), P is a state transition function, R is reward function and γ is the discount factor.

- **State:** A state is an intermediate graph represented by G^τ
- **Action:** An action at any time τ, is give by $a^\tau = (\xi, u, v, b)^\tau$. If at any time step $\xi^\tau = 0$, then the reaction is considered terminated, i.e, that action and all subsequent actions are ignored. While performing actions, valency checks are not performed as an intermediate invalid graph may still lead us to the correct final graph.
- **Reward:** Both intermediate and final rewards are given. Intermediate rewards are given based on correct sub-actions and the final rewards is dependent on the final product.

The workflow for Do *et al.* [2019] is summarised in Figure Figure 2.2 Though [Do *et al.*, 2019] is proposed for chemical reactions, it can easily be extended to other graph transformation problems.

Paper Summary: A Generative Model For Electron Paths

Bradshaw *et al.* [2019] propose a reaction prediction framework for linear electron flow(LEF) reactions [Herges, 1994]. Though this reduces the USPTO dataset's size by 30%, it allows for a lighter model in comparison to [Jin *et al.*, 2017*a*; Do *et al.*,

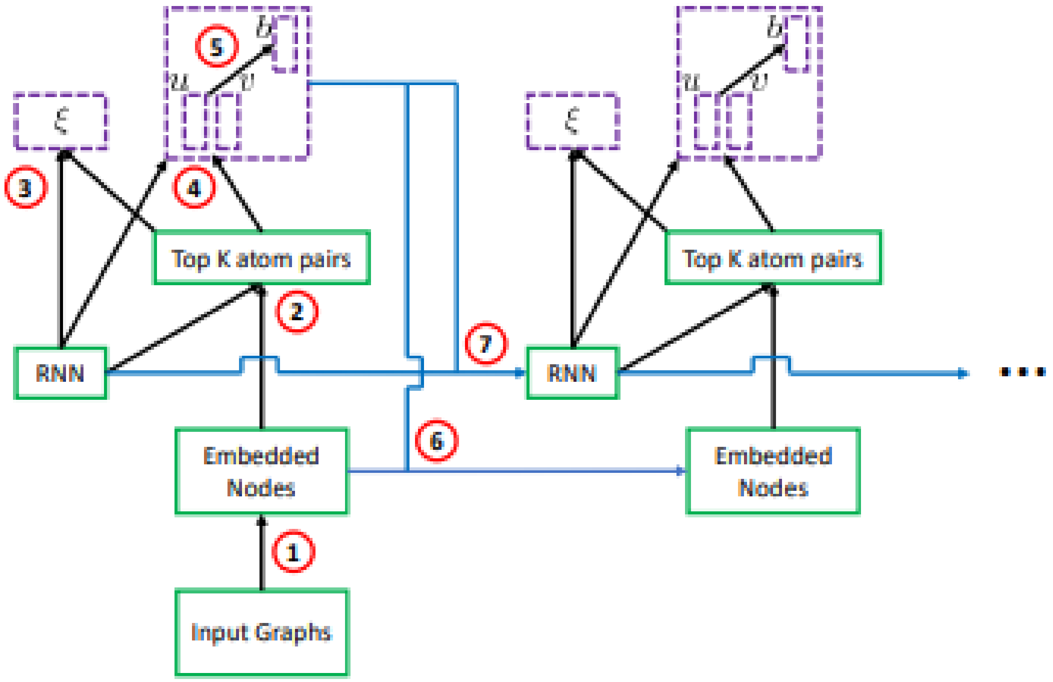

Figure 2.2: A pictorial summary of Do *et al.* [2019]

2019] which can be trained in just 10 epochs. This paper also separates the reactants from the reagents as a pre-processing step. This is in contrast to other papers using this dataset.

[Bradshaw *et al.*, 2019] models LEF reactions as a sequence of electron movements. They model the transformation of reactant (G_0) to product (G_T) as a sequence of electron movements. Alternate steps will add and remove electrons from different atoms. At each stage, the model decides whether the reaction continues or stops. If the reaction continues, then based on the previous action taken, the model picks an atom to add or remove electrons from.

Table 2.1: Results of [Jin *et al.*, 2017*a*; Do *et al.*, 2019; Schwaller *et al.*, 2018; Coley *et al.*, 2019]

Paper	Top-1	Top-3	Top-5
Jin *et al.* [2017*a*]	79.6%	87.7%	89.2%
Schwaller *et al.* [2018]	80.3%	86.2%	87.5%
Do *et al.* [2019]	83.2%	86.0%	86.48%
Coley *et al.* [2019]	**85.6%**	**92.1%**	**93.4%**

Results of mentioned papers

The results are evaluated on a test set consisting of 40K reactions. Evaluation metric used is top-K accuracy. Since Bradshaw *et al.* [2019] works only on LEF reactions, and requires separation of reactants from reagents, it cannot be compared fairly to other models.

Retrosynthetic Planning

The problem of retrosynthetic planning takes a molecule as input, and gives as output a set of molecules which could potentially react to give that molecule. It differs from the forward prediction problem as it gets only a subset of products as input, and the solution may involve other by-products. One approach to the retrosynthesis problem involves searching through a reaction network as is the case in [Fagerberg *et al.*, 2018; Segler *et al.*, 2018]. There are also commercial softwares such as Chematica [Grzybowski *et al.*, 2018] which take this approach. Such an approach to building a synthesis planner is time consuming and limited to the encoded knowledge base, as can be seen in Chemtica which took more than a decade to build.

Baylon *et al.* [2019] classifies the product molecule based on which reaction template could have produced it. Liu *et al.* [2017] proposes a template free seq2seq model for retrosynthetic planning, and reported results similar to template based approaches.

2.2 Molecule Generation

The space of drug-like molecules is estimated to contain 10^{60} molecules [Virshup *et al.*, 2013]). Existing datasets of drug-like molecules contain a very small subset of this space. The process of drug discovery relies heavily on existing datasets of drug-like molecules. In order to expand the search space of drug design, we can make use of generative models. These models can also be optimised to generate molecules with certain desired properties. This can help us search the space of molecules in a much more efficient manner.

In this section, we will present a brief summary of the use of deep generative models for molecule generation. These deep generative models include RNNs, VAEs, GANs, and adversarial autoencoders [Makhzani *et al.*, 2015]. Our purpose here is to give the reader a flavour of the various methods and tasks which are being solved using generative models for molecules. For a more complete review of such models, the reader is referred to [Xu *et al.*, 2019; Elton *et al.*, 2019]

2.2.1 Datasets

The commonly used datasets for the purpose of molecule generation are presented in Table 2.2.

Table 2.2: Commonly used molecular datasets

Dataset	Molecule type	No. of molecules
ChemBL [Gaulton *et al.*, 2016]	Bioactive drug-like small molecules	1.8 million
ZINC [Irwin and Shoichet, 2005])	Commercially available compounds for virtual screening	230 million
QM9 [Ramakrishnan *et al.*, 2014]	Up to nine heavy atoms	134K

Apart from these, there also exist some other molecule databases. Of particular interest is the GDB-17 dataset [Ruddigkeit *et al.*, 2012], which enumerates 166 billion organic molecules of up-to 17 heavy atoms.

2.2.2 Evaluation Metrics

Evaluation for molecule generation can be fairly tricky. A good generative model needs to be able to produce valid molecules as well as mantain diversity in generated molecules. It also needs to be able to produce novel molecules, which differ from the training dataset. However, the molecules produced should share some properties with the training set. Keeping these things in mind, the commonly used evaluation metrics are novelty, validity, uniqueness, and diversity. Consider a generative model trained using a set of molecules denoted by T. This model is used to generate n_s molecules. The valid molecules generated is denoted by V, and the unique valid molecules is denoted by V_u. To measure similarity between two molecules the Tanimoto co-efficient(T_d) on extended connectivity fingerprints is commonly used [Rogers and Hahn, 2010]. The Tanimoto co-efficient is equivalent to the Jaccard Index, as defined in Table 5.2.

$$Validity = \frac{|V|}{n_s} \tag{2.8}$$

$$Novelty = 1 - \frac{|V_u \cap T|}{|V_u|} \tag{2.9}$$

$$Uniqueness = \frac{|V_u|}{|V|} \tag{2.10}$$

$$Internal\,Diversity = \frac{1}{|V_u^2|} \sum_{(x,y)\in V_u} T_d(x,y) \tag{2.11}$$

$$External\,Diversity = \frac{1}{|V_u||T|} \sum_{x\in V_u,\, y\in T} T_d(x,y) \tag{2.12}$$

Further, Benhenda [2017] propose the ChemGAN diversity challenge, which requires generative models to have an internal diversity greater than or equal to the internal diversity of the training set of molecules. This challenge maybe a bit too restrictive. A better metric along similar lines would be the ratio between internal diversity of generated set and internal diversity of training set.

2.2.3 SMILE Based Approaches

Here, we present a brief overview of deep generative models for molecule generation, which use the SMILE representation. Most SMILE based generative models try to generate molecules in a sequential manner. They one hot encode SMILES characters, and try to learn them, similar to learning generative models for text.

Segler *et al.* [2017] uses an LSTM [Hochreiter and Schmidhuber, 1997]. The LSTM is trained using the ChemBL dataset first. This model can be used to generate molecules similar to the ones it is trained on. The authors also fine-tune the model by training it on a set of molecules which are known to have some affect on the Malaria virus. Before training, the Malaria dataset is split into training and hold-out. The hold-out data has 1240 molecules. The model is used to generate 128K molecules. The authors report that the generated molecules contained 14% of the held out data.

[Olivecrona *et al.*, 2017; Jaques *et al.*, 2016] are similar to that of Segler *et al.* [2017]. These paper also proposes using an RL network on top of the existing LSTM network to generate molecules with some desired properties. [Gómez-Bombarelli *et al.*, 2018] propose using a VAE to represent molecules in a continuous space. This approach not only allows them to generate new molecules, but also optimise molecules using Gaussian processes [Rasmussen and Williams, 2005] by traversing through the continuous representation of molecules. Further, they can also use the continuous representation of molecules for QSAR models. Another paper which follows a similar approach is Blaschke *et al.* [2017]. The only difference is that Blaschke *et al.* [2017] uses an adversarial autoencoder instead of a VAE. Dai *et al.* [2018] propose a context free grammar for generating SMILES. Their model ensure checking of syntax and semantic rules. They train a VAE based model to generate molecules which learns under the restrictions of the grammar. Yu *et al.* [2016] proposes SeqGAN which is a framework for generating sequences with policy gradient. The paper by-passes the problem of gradient update for generators(for discrete data) in GANs, by directly performing gradient policy update on the generator. Guimaraes *et al.* [2017] propose an extension of SeqGAN titled ORGAN. ORGAN can optimise on any desired property for the molecule.

2.2.4 Graph Based Approaches

Here, we will cover papers which rely on the graph representation of molecules for generation. Samanta *et al.* [2018] propose NeVAE, which is a VAE based model for graph generation. In NeVAE, the encoder uses a GCN to learn a vector representation of the molecule. This feature vector is passed through a neural network to ensure that the resulting output follows a standard normal distribution. The decoder, first samples from a distribution the number of atoms to be generated. For each node it samples which node type it is, i.e a feature vector for the node. Based on this feature vector, it calculates the probability of various bond types between all pairs of nodes. NeVAE uses masking to ensure the generation of valid molecules. NeVAE can also search through the latent space using Bayesian optimisation to find molecules which have desired properties.

Jin *et al.* [2018] propose JT-VAE, which decomposes a molecule using the junction tree algorithm. The junction tree algorithm is used to decompose a graph into a tree scaffold. The generative model works in two steps. It first generates the tree scaffold and node labels. Then, it labels the various nodes and tries various connectivity variations, scoring them based on difference from the starting graph. For the junction tree algorithm to operate, a vocabulary of atom clusters is learned. Then using sub-graph matching between the vocabulary and the molecule, the molecule is decomposed into a junction tree. JT-VAE can also seek molecules optimising on a specific property globally or locally, i.e in the neighbourhood of a given molecule.

Simonovsky and Komodakis [2018] propose GraphVAE, which generates molecules in a one shot manner using a VAE. GraphVAE outputs a fully connected probabilistic graph. However, this approach is constrained to small molecules.

The paper sticks to molecules containing 9 heavy atoms. Further, the model architecture needs to be designed keeping in mind the maximum molecule size in the dataset.

Liu *et al.* [2018*a*] propose a variation of graphVAE. Instead of predicting a fully connected adjacency matrix, this model predicts the edge labels in a sequential manner. Cao and Kipf [2018] propose MolGAN, which is a GAN based approach to generating molecules. MolGAN consists of a discriminator, generator and reward network. The generator generates a graph from a random vector. The generator outputs a fully connected probabilistic adjacency matrix. The discriminator attempts to discriminate between real and fake inputs. The reward network rewards an input molecule based on a desired property.

You *et al.* [2018*b*] proposes GraphRNN, which generates a graph in a sequential manner. The key driver of the paper is a sequential representation of the graph. At the time of generation of each node, the node is classified, and edge probabilities between the new node and previously generated nodes are predicted.

You *et al.* [2018*a*] proposes graph convolution policy network (GCPN) for molecule generation. GCPN is a reinforcement learning framework for molecule generation. At each stage of the generation process, the action taken can be adding a new cluster (clusters maybe taken as individual atoms or similar to JT-VAE) maybe connected to the existing graph, or a new edge maybe added between existing clusters.

Finding the right drug for a disease is like finding a needle in a haystack. While we cannot replace the entire drug design process with deep generative models, deep generative models can help us search the molecular space in a much more structured and efficient manner.

2.3 Deep Learning on Networks for QSAR

QSAR or quantitative structure activity relation models are models which predict activities or properties of chemicals using just the chemical structure information. Typically QSAR models rely on molecular descriptors as feature vectors, on which a classification or regression model is learned. This model maybe a simple machine learning model or a deep learning model. Our focus here, will be on the application of deep learning to learn these feature vectors in an automated fashion using deep learning.

Goh *et al.* [2017*b*] convert a 2-D representation of a molecule into an image and make use of CNNs for QSAR prediction. With this simple approach they are able to achieve decent results.

Gómez-Bombarelli *et al.* [2018] proposed a differentiable molecular fingerprint for end-to-end learning on QSAR tasks. They draw inspiration from the commonly used, but non-differentiable circular fingerprints for molecules. The algorithm they propose for building the feature vector is in essence a GCN. They demonstrate improved results on a variety of tasks such as solubility and drug efficacy from the use of differentiable fingerprints as compared to standard circular fingerprints.

Li *et al.* [2017] use GCNs along with a super-node to learn graph level features to achieve state of the art results on a variety of datasets including the Tox21 dataset. For predicting any property of a molecule, the model needs to learn a vector representation of the molecule from node features. Prior works sum up node features to learn a graph level representation of the molecule. In this paper, the authors make use of a super node and learn a weighted sum of individual nodes to learn a better graph level representation.

Li *et al.* [2018] introduces a graph learning metric prior to applying GCNs on the molecular network. They propose a new spectral convolutional layer, which allows them to learn a distance metric for the graph, depending on the task at hand.

While there is some work on automatically learning molecular features using GCNs, there is a need for a comparative study of various GCN architecture variants. This could include variation of depth of GCN, graph pooling layers, variations of atom and bond features, among other things. Another interesting research direction in this field could be for the identification of cliff points. This boils down to removing one atom from the molecule and identifying the variation of various molecular properties. Significant changes due to small variations of molecular properties can help in identifying important functional groups.

CHAPTER 3

Case Study: Reaction Prediction Using GCN on USPTO Dataset

In 2.1 we looked at a survey of methods being applied for chemical reaction prediction. In this chapter we will take a deeper look into the USPTO dataset. Further, we will analyse the works of [Jin *et al.*, 2017*a*; Coley *et al.*, 2019] and suggest improvements for them. We present state of the art results on this dataset, even though our model has 25x lesser training parameters than the prvious baseline [Coley *et al.*, 2019].

3.1 USPTO Data Analysis

We present an analysis of the USPTO dataset. This would help us get an idea of the size of the data, and the various neural net architectures we can use to attack this problem.

The dataset consists of 479,035 reaction samples. In Jin *et al.* [2017*a*] the dataset is split into training(409,035), validation(40K) and test(30K) samples. We follow the same split to compare results. Each reaction sample consists of the reaction SMILE, and the edits needed to re-create the reaction sample. An edit is is a three tuple representation of changes in bond order. It contains the two atom ids between which there is a change in bond order in the course of the reaction, and the bond order in the product. Next, we take a look at the distribution of the size

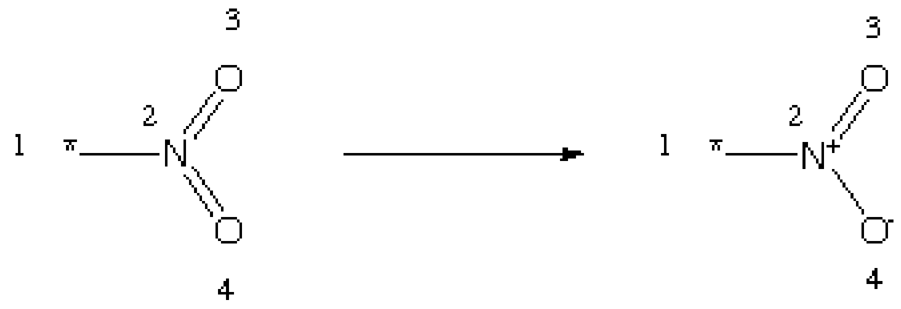

Figure 3.1: A sample reaction.

The corresponding reaction SMILE is [C:1][N:2](=[O:3])=[O:4]>>[C:1][N+:2](=[O:3])[O-:4]. There is only one bond order change in the above reaction, and the corresponding edit is (2,4,1.0)

of the reactions, i.e the number of components in the reactions.

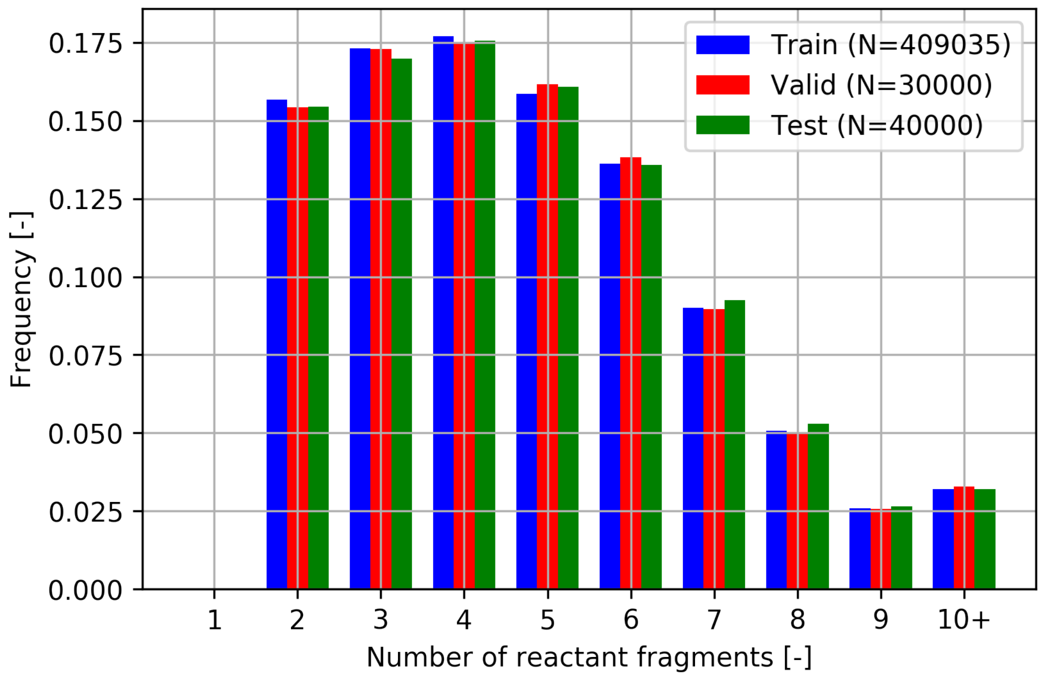

Figure 3.2: Distribution of number of fragments in the reactants

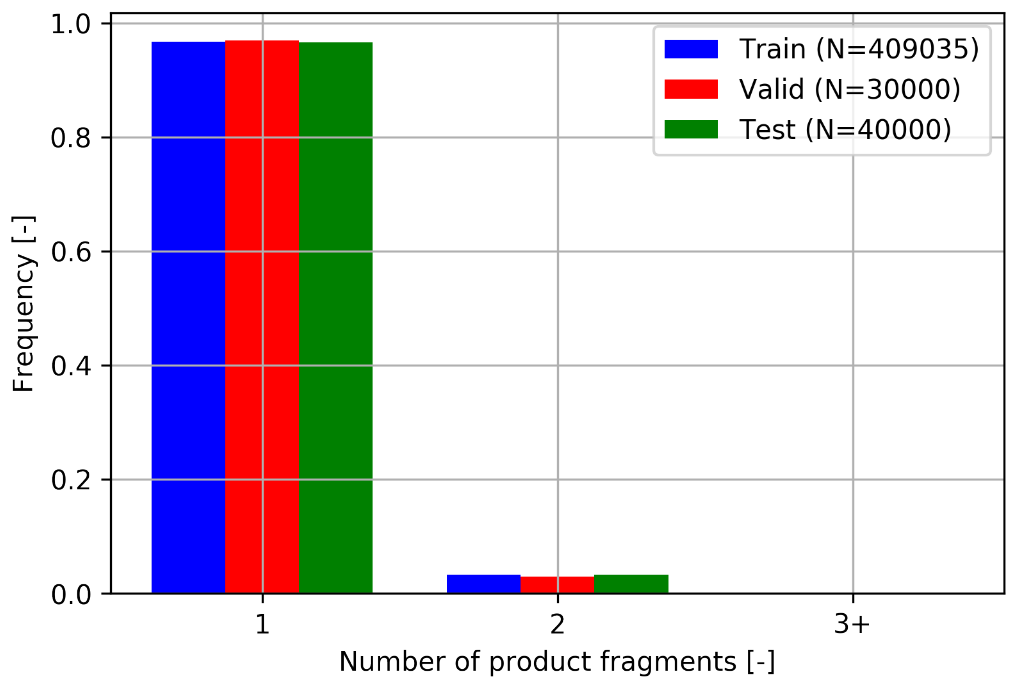

Figure 3.3: Distribution of number of fragments in the products

There are two reasons for the large differences between Figure 3.2 and Figure 3.3. The first reason is that a large number of reactions are combination reactions and lead to a reduction in the number of components. The second reason is that products do not contain reagents and catalysts. These are molecules which though present in the reactants, do not undergo any transformation themselves in the course of the reaction.

Next we look at the distribution of number of atoms in the reactants and products.

As we see from Figure 3.4, the maximum number of atoms in the reactants are 150. Thus the maximum size of the adjacency matrix can be 150. However, we also see that majority of the reactants contain less than 60 atoms, and keeping the adjacency matrix to contain 150 nodes for all cases will be an overkill. Instead, we bin the reactions based on the number of atoms it contains, and in each training

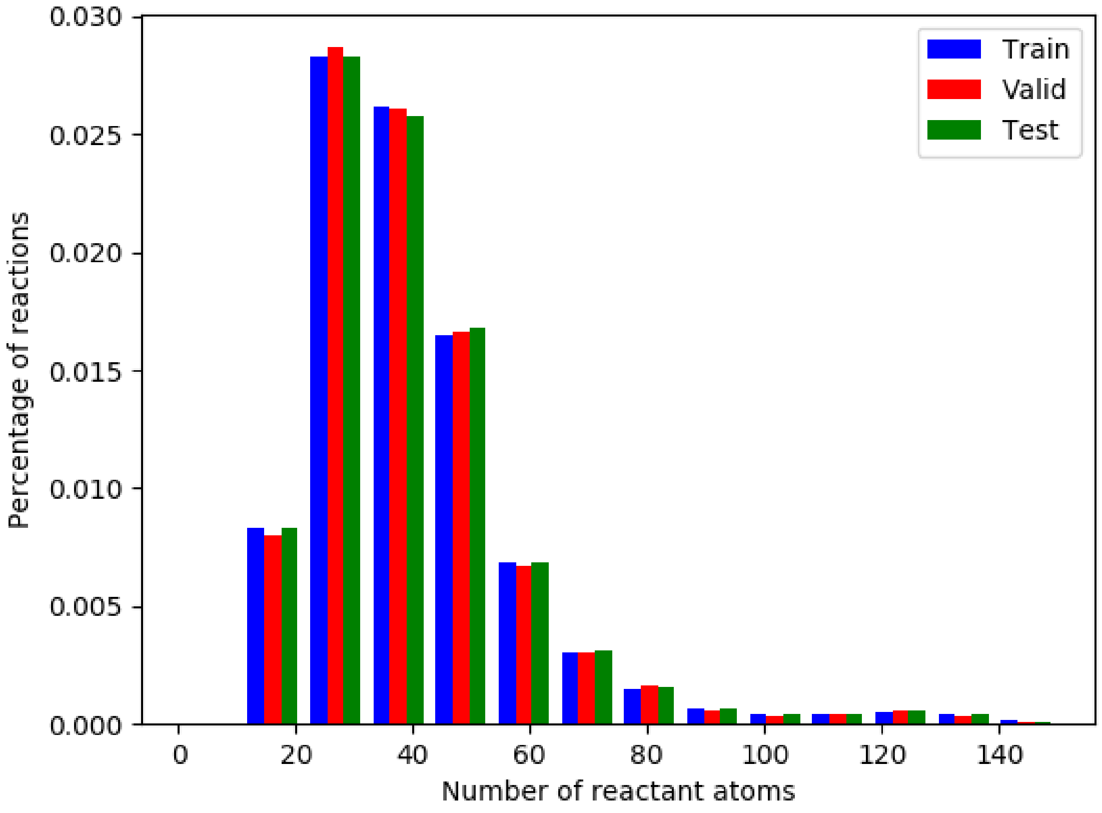

Figure 3.4: Distribution of number of atoms in the reactants

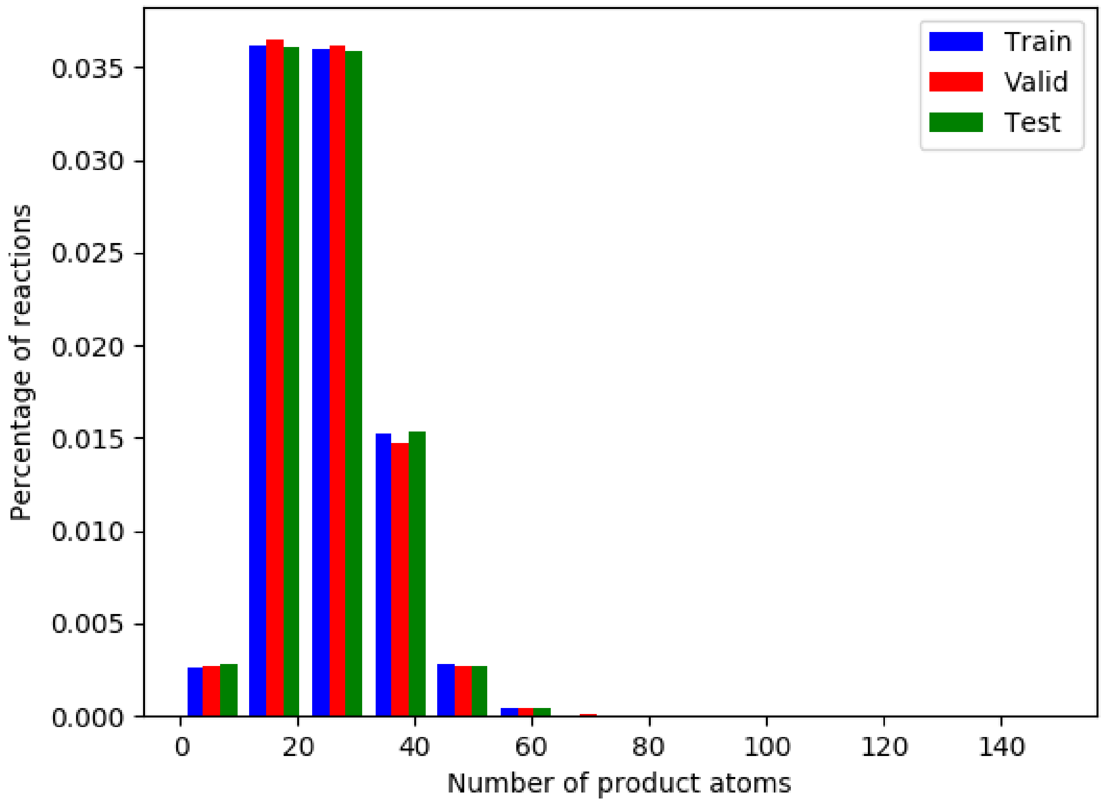

Figure 3.5: Distribution of number of atoms in the products

epoch pick reactions from one bin.

Next, we look at the distribution of number of changes in a reaction. This is depicted in 3.8. We see that the maximum number of changes in any reaction is 6.

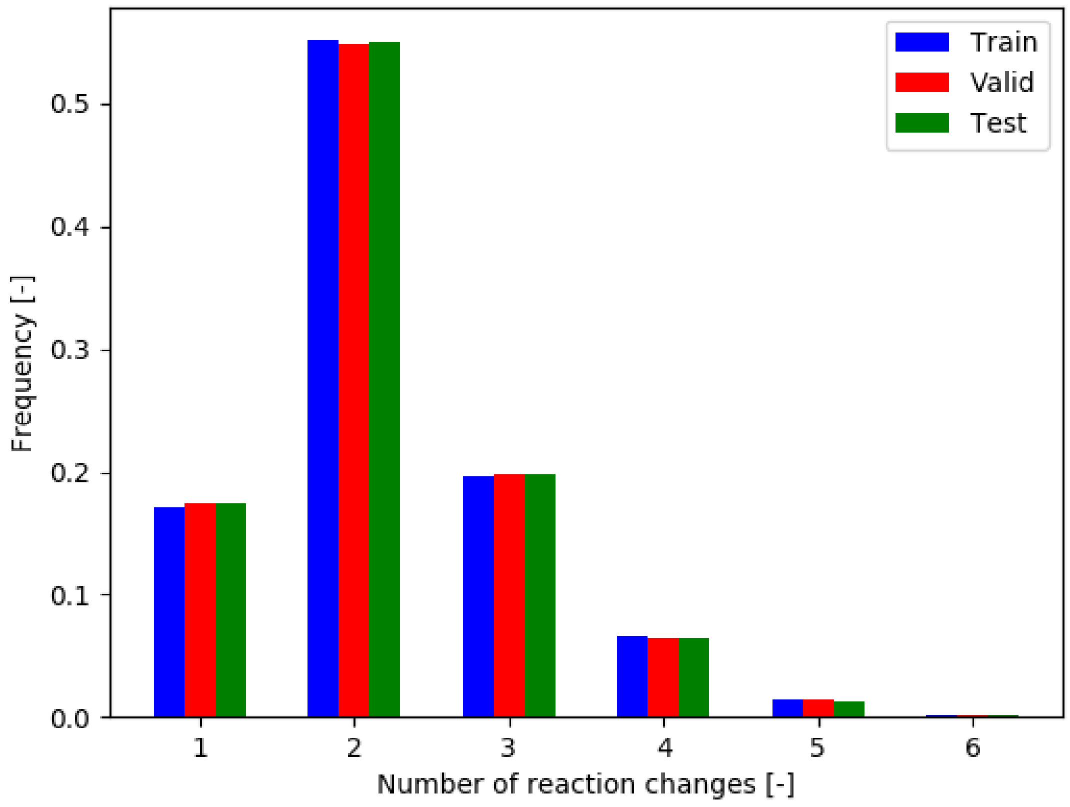

Figure 3.6: Distribution of number of edits

This is a very small number of reactions, and we can choose to restrict ourselves to reactions with 5 changes as the number of reactions with 6 changes is very small ($< 0.2\%$).

Next, we look at the various atoms present in the reactants and products.

We find that there are 63 types of atoms present in the dataset. To represent atom type, we append a one-hot encoding of the atoms to the node features. We also see that the most common types of atoms are carbon, oxygen and nitrogen.

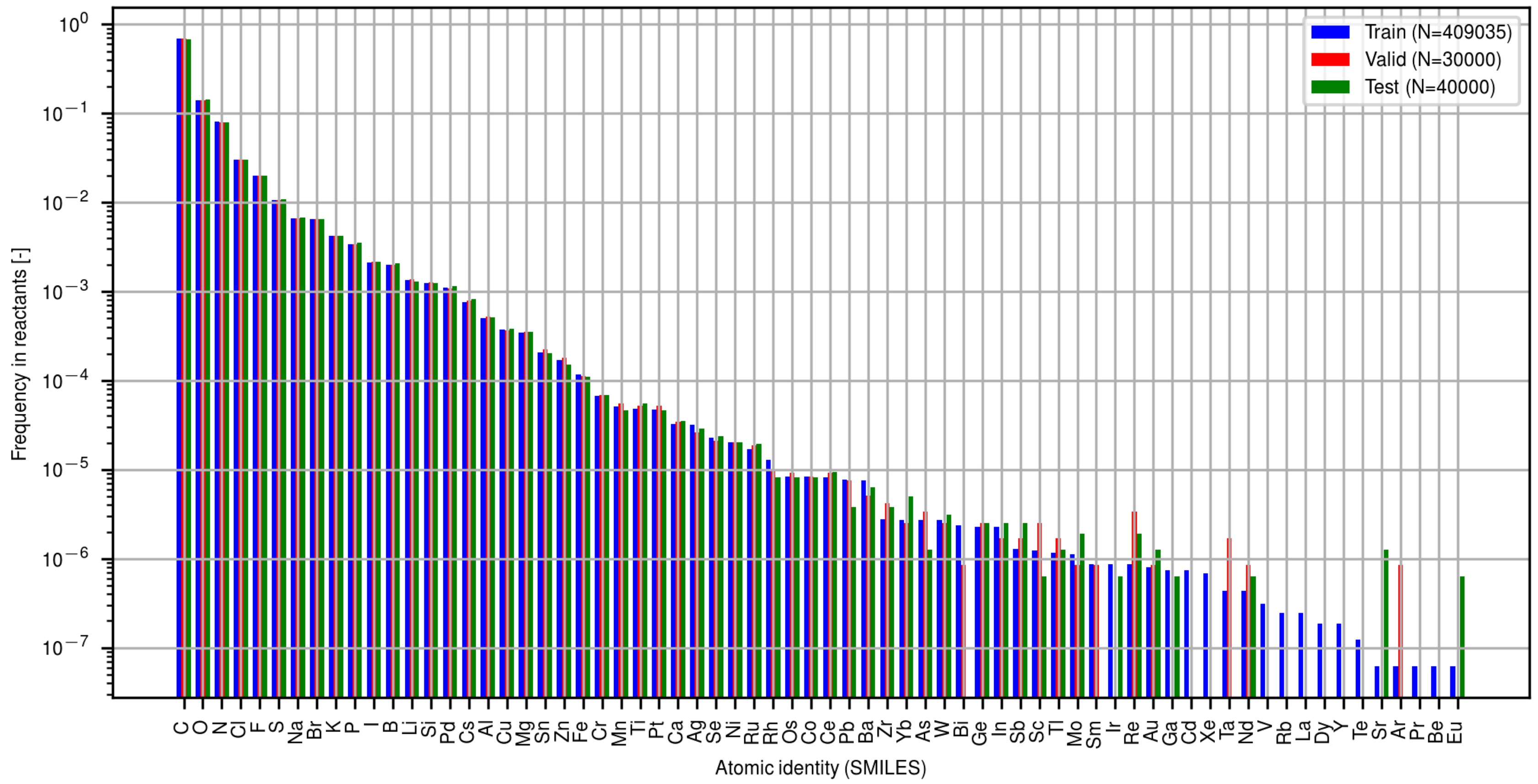

Figure 3.7: Number of atoms of various types in the reactants

This is to be expected, as we are dealing with organic reactions.

3.2 Identification of Candidate Edits

The works of [Jin *et al.*, 2017*a*; Coley *et al.*, 2019] split the reaction prediction problem into two parts. The first part requires scoring each possible edit during the reaction. The second part takes the top-K edits from the first model, and runs a combinatorial search through them to obtain a set of candidate products and identifies the most likely candidates. Despite the drawbacks of this approach, it is the current state of the art.

We present an analysis of the first part in this section, and the second part in the next section. We already presented an overview of [Jin *et al.*, 2017*a*; Coley *et al.*, 2019] in subsection 2.1.

3.2.1 Effect of Global Features

We first analyse the effect of global features on the model. The global features of the graph are aggregated according to Equation 2.4.

The evaluation metric used is coverage. Coverage at some K is the percentage of reactions for which the reaction edits are a subset of the top-K candidate edits.

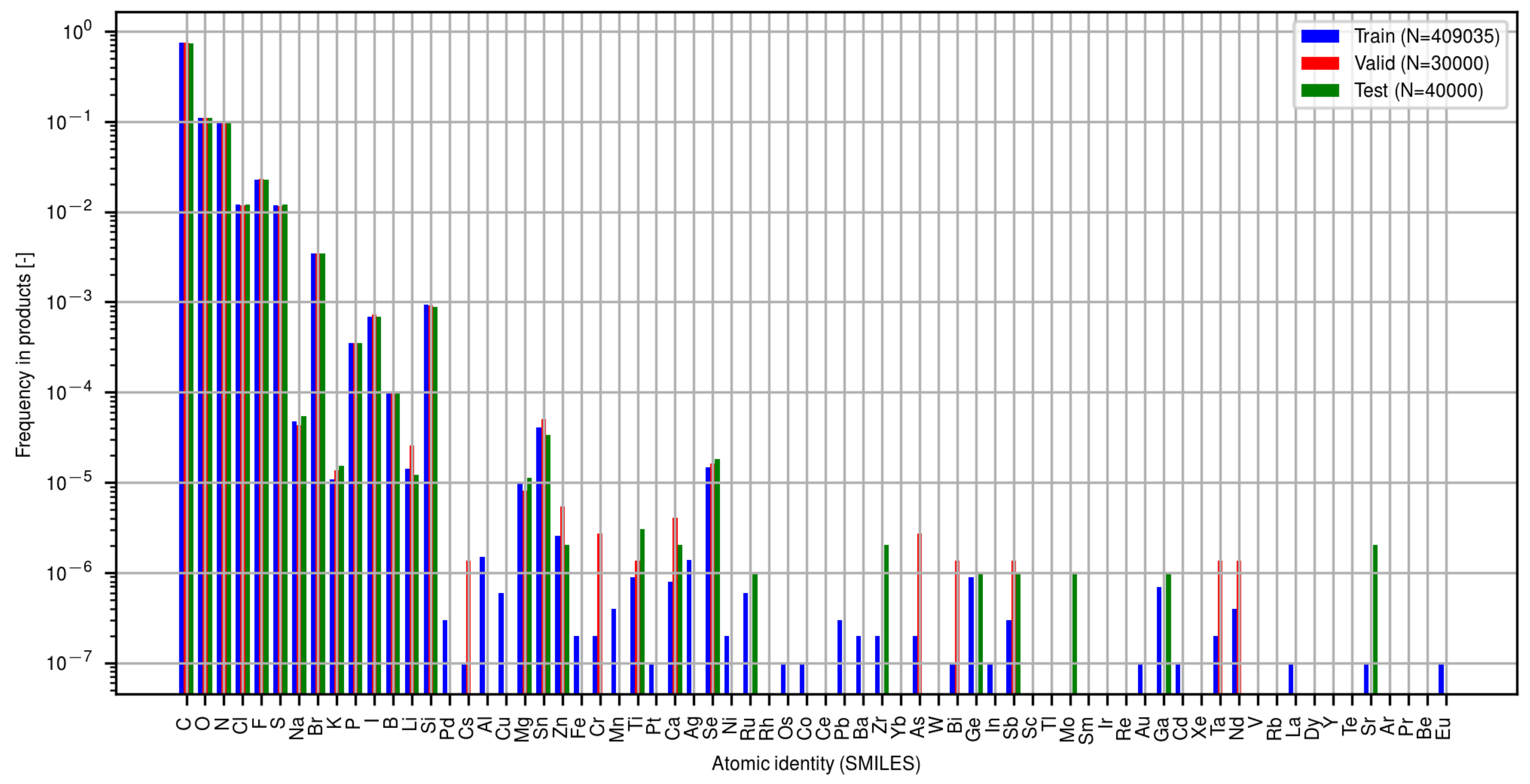

Figure 3.8: Number of atoms of various types in the products

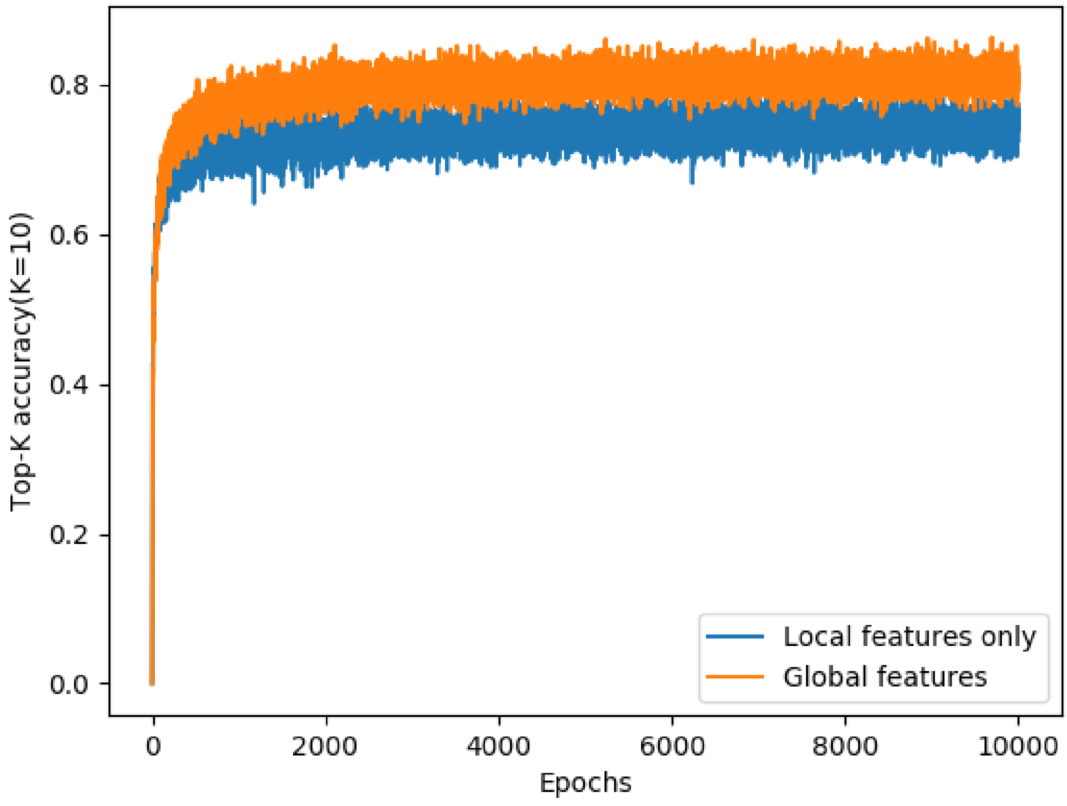

Figure 3.9: Effect of global features on training accuracy

We find a drop of around 10% in accuracy while using only local features.

3.2.2 Effect of Fusion GCN

We propose two changes to the original model of Jin *et al.* [2017*a*]. Firstly, we do away with the final layer proposed by them as per equation 2.2. This is because carrying out this operation leads to a significant increase in the number of parameters and loss of information. It requires the network to broadcast both the edge information and node information to the same dimension which is very inefficient as the nodes contain much more information than the edges.

Secondly, we add a fusion component to the network. Vijayan *et al.* [2018] propose a fusion GCN which allows the network to regulate information flow from different depths. The fusion operation can be represented by

$$c_v^N = \sum_{i=0}^{N} W_i c_v^i \tag{3.1}$$

We find that these changes allow us to improve on the baseline.

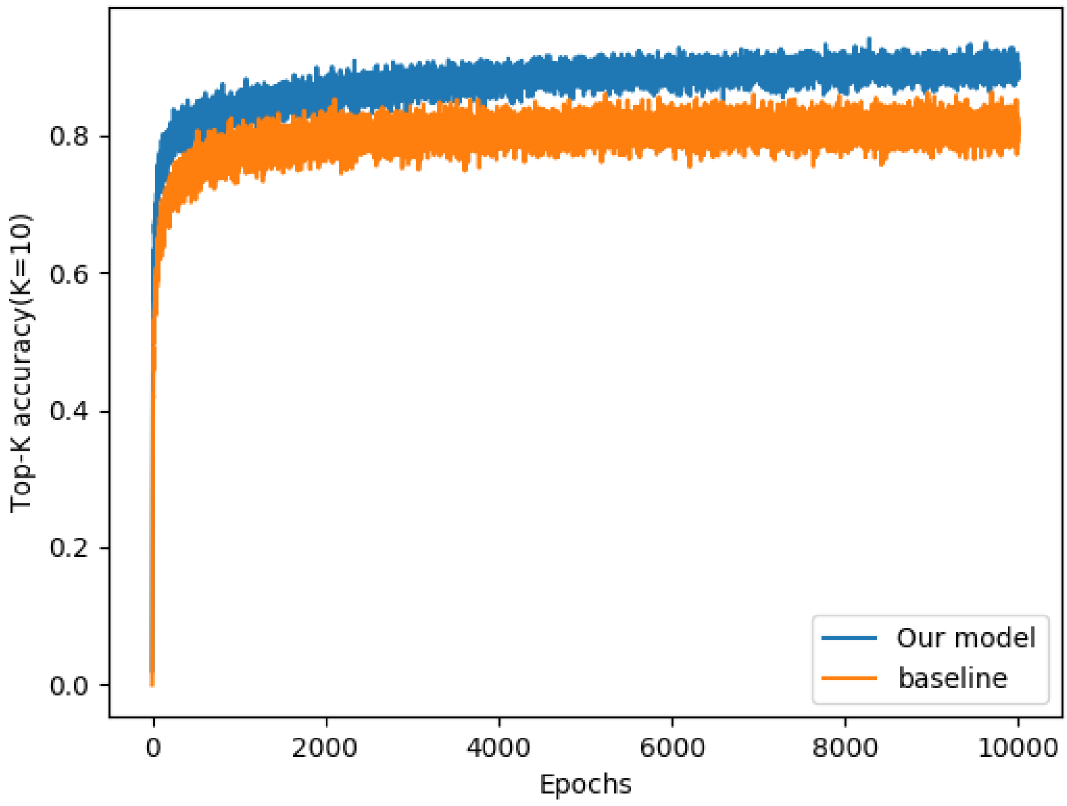

Figure 3.10: Effect of fusion on training accuracy

Table 3.1: Comparison of coverage results for test data

Model	Acc@7	Acc@10	Acc@12	Acc@16	Acc@20	Acc@80
Coley *et al.* [2019]	72.8%	84.7%	86.9%	90.4%	91.9%	97.1%
Our Model	**74.5%**	**87.9%**	**87.9%**	**90.9%**	**92.3%**	**97.1%**

We present the top-K accuracy of reaction centre identification in table 3.1. We count a reaction as correctly identified if all correct edits are present in the top-K edit predictions.

3.2.3 Effect of Depth

Next, we look at the effect of the depth of the GCN. All above mentioned experiments were carried out for a depth of 2.

We carried out experiments for depths 1,2 and 3. We did not find any significant improvement when we increased the depth from 2 to 3. Our results are shown in 3.11. From this we can say that it is optimal to choose a depth of 2 for the model.

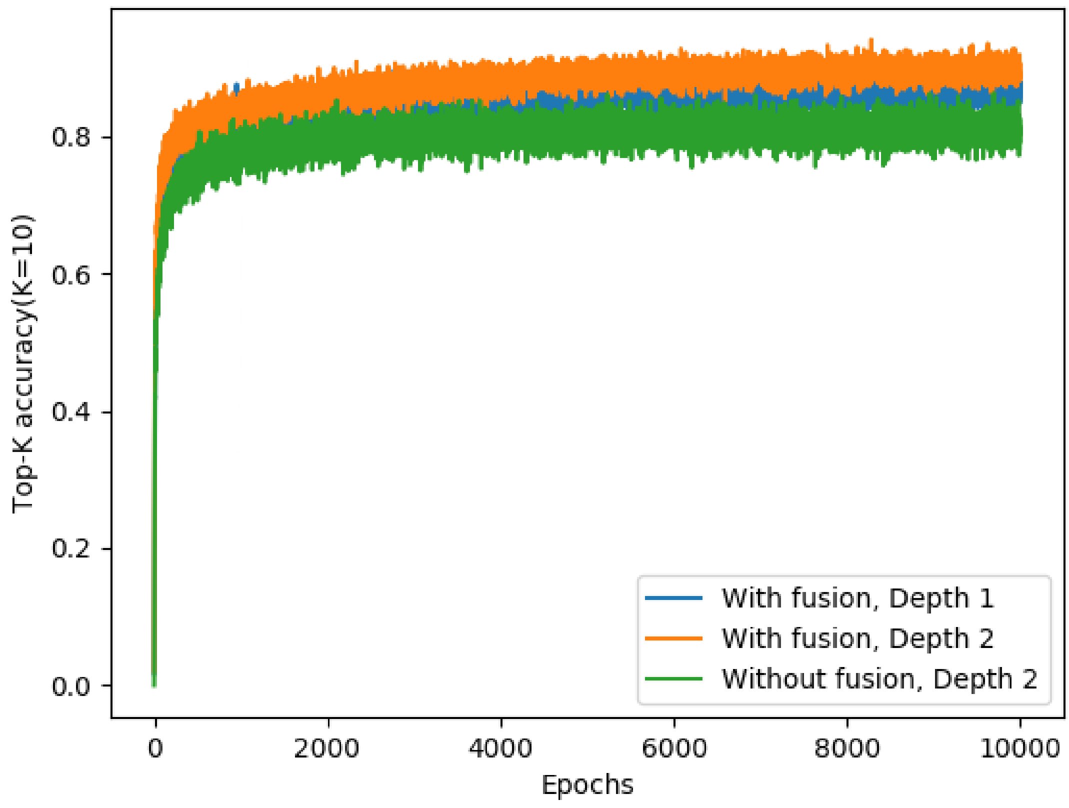

Figure 3.11: Effect of depth on training accuracy

3.2.4 Effect of Additional Node Features

Typically the model takes the following features:

- Atom type
- Explicit Charge
- Valency
- Aromaticity

We try to add the following node features as well to see if it improves model performance:

- Chemical group
- Partial charge
- Aromaticity of neighbours
- Whether the atom is in a ring

Our results are shown in 3.12. We do not find any significant improvements by using these additional features.

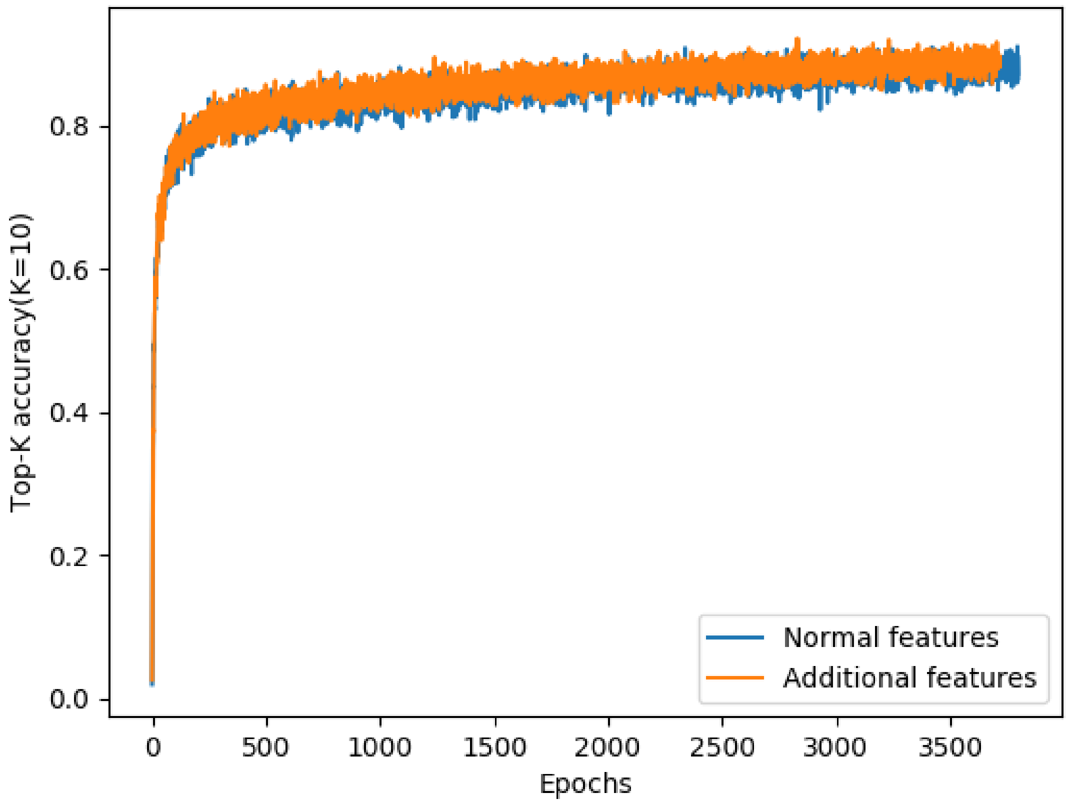

Figure 3.12: Effect of additional features on training accuracy

3.3 Predicting Reaction Products from Candidate Edits

The first part of the model scores each possible edit for the reaction. We now take the top-K(K = 16 for sake of comparison with Coley *et al.* [2019]) edits based on the score and run a combinatorial search of subsets of upto size 5. Each subset gives a candidate product. The invalid candidates are discarded and the valid ones are scored using 2.7.

The baseline model makes use of an atom embedding size of 500, which leads to a fairly large model. The large size of the model, in combination with the combinatorial search needed make the second part of the model somewhat difficult to train. We reduce the embedding size for our model, cutting our training time by a factor of 5, as a result.

We present the accuracy results of the reaction prediction problem in 3.2. We count a reaction as correctly predicted if the correct product is present in the top-K predictions.

While we achieve only a small improvement in comparison to the baseline, our model is 25 times smaller and much easier to train.

3.4 Conclusion and Future Work

We find that using a more advanced GCN architecture allows us to learn faster and more accurately. We were not able to experiment with a large number of GCN architectures due to long training times. To avoid this problem, there is a need for

Table 3.2: Comparison of accuracy on test data

Model	Model Size	Embedding Size	Top-1	Top-3	Top-5
Coley *et al.* [2019]	1800K	500	85.6%	92.1%	93.4%
Coley *et al.* [2019]	72K	100	54.4%	73.1%	79.4%
Our Model	72K	100	**86.4%**	**92.3%**	**93.5%**

a smaller dataset for quick prototyping of an efficient architecture, which can be later tested on the complete dataset.

The current research for reaction prediction relies heavily on the USPTO dataset. However, this dataset does not contain any information regarding physical conditions and reagents/catalysts needed. There is a need for a dataset which contains this information.

While the current models which solve this problem are powerful, there is a lot of scope for improvement in terms of training time as well as accuracy. Here, we have demonstrated that it is possible to do both simultaneously. It is also possible to do both these things by considering an entirely different pipeline as some recent works have done, which we discussed in Chapter 2. However, none of them are able to match the accuracy of our model. Furthermore, these papers do not report the time taken and hardware used to train the model, making it difficult to compare the efficiency of such models.

CHAPTER 4

Survey of Deep Learning Applications in Network Biology

In Section 1.3 we introduced the various types of networks commonly present in biology. Here, we focus on the application of deep learning to these networks.

4.1 Protein-Protein Interaction Networks

Conventional methods for identifying protein-protein interactions are either experimental in nature or are based on co-functionality or co-evolution. For a review of conventional methods used to identify protein interactions the reader is referred to Raman [2010].

However, the network built by such methods is noisy and incomplete. Computational methods can help us not only fill in the gaps in the network, but also help us extract valuable insights from the network.

We focus our attention on the application of deep learning to PPI networks, where the task at hand is either node classification (identification of function) or link prediction (identifying new protein interactions).

4.1.1 Identifying Protein Functions Using PPI Networks

Hamilton *et al.* [2017*a*] introduce a new dataset for classification of proteins based on their cellular functions. The dataset consists of 24 graphs, each corresponding to a different human tissue. They propose the use of 20 graphs for training, 2 for validation and 2 for testing. The same split has now become commonplace [Vijayan *et al.*, 2018; Liu *et al.*, 2018*b*] for the task of function classification on PPI networks. The general approach followed for this task is to aggregate network information to the node attribute information using GCNs and using these aggregated node attributes for node classification to identify protein function.

Most works use positional gene sets, motif gene sets and immunological signatures as features and gene ontology sets as starting node attributes for the node classification task. While this approach is powerful, it would be interesting to try a richer set of starting features such as sequence information.

Zitnik and Leskovec [2017] proposes OhmNet, which makes use of multi-layer networks for identification of tissue specific functionality of proteins. Each layer contains information regarding a different tissue. OhmNet encourages sharing of information across layers through the multi-layer network. This is an interesting direction of research, which leverages the fact that the same protein can be present in different tissues, and different information regarding it can be extracted from each tissue's PPI network.

4.1.2 Link Prediction for PPI Networks

PPI networks are highly modular in nature and links are highly affected by the local network structure. This is because of the formation of functional clusters in PPI networks, i.e proteins which are functionally related will form a cluster in the graph. Most PPI data available to us is incomplete and noisy. Computational methods can help in building a more accurate PPI network. Conventional methods have been applied to the problem of link prediction [Hulovatyy *et al.*, 2014; Lei and Ruan, 2012]. While these methods are of significant value, our focus is on the application of deep learning to this problem.

Zhang and Chen [2018] proposes the use of graph neural networks for link prediction, treating PPI network as a case study. They delete half the existing links from the graph and attempt to predict the deleted links. Similar approaches using GCNs have been attempted as well for a variety of PPI datasets.

Khan *et al.* [2018] rely on protein sequences to determine binding affinity of a pair of proteins. It applies a sequence of convolutional layers on the sequence of amino acids to obtain a vector representation of a protein. Then, it takes the hadamard product of the vector embeddings before training a classifier for learning protein links. Such an approach, however, fails to capture the inherent network information present in the dataset.

Richoux *et al.* [2019] compares deep learning based methods which rely on the sequence information of proteins. They take extra care to prevent data leaks between the train and test data by ensuring that no common proteins exist in the two sets. However, they also fail to utilise network information in their approach.

The current approaches for link prediction for PPI are fairly basic in nature as most papers dealing with this problem treat it only as a case study. The problem of link prediction in proteins is a fairly complex one, and requires dedicated algorithms and research. Further, there needs to be research in the direction of efficiently building node and edge attribute information for the PPI prediction problem.

4.2 Metabolic Networks

All cells are able to perform their functions through a series of chemical reactions. The substances directly or indirectly participating in these reactions are called metabolites. Generally, the product of one reaction acts as the substrate for the next reaction. The set of chemical reactions that occurs in the cell can be modelled as a network. This network of chemical reactions can be used to identify various metabolic pathways, which in turn can help us gain insight into the working of cells, and identify crucial reactions and metabolites.

Sankar *et al.* [2017] propose modelling metabolic reactions as a transformation of molecular networks, and build a model to predict chemical reactions. Dale *et al.* [2010] propose a model for identifying metabolic pathways using a data-driven approach.

4.3 Drug Target Interaction

The interaction between drug and target is modelled using 3-D molecular simulation techniques such as docking at the virtual screening to identify potential drug candidates. However, docking is a computationally expensive technique. Further , such methods cannot be applied in case the 3-D structure of the protein is not known. Recently, attempts have been made to apply deep learning methods to compute binding affinity between proteins and drugs.

Various computational approaches have been tried in order to identify positive interactions between a set of molecules and protein targets. One simple method proposed by Keiser *et al.* [2007] is to use similarity to known ligands to identify potential ligands. However, such an approach will fail if the key functional group is the only difference between two ligands. Prado-Prado *et al.* [2011] pose the problem of identifying drug target binding affinity as a QSAR problem, generalised across proteins and ligands. They build 3D structure features for proteins and ligands, and use these features to train a classifier, which learns the binding affinity between the molecules. Tian *et al.* [2016] uses a similar approach, but uses an Artificial neural network (ANN) to learn the classifier. Öztürk *et al.* [2018] propose the use of ligand and protein sequence information to predict binding affinity between them. Unlike previous papers, Öztürk *et al.* [2018] does not treat the prediction of binding affinity as a binary classification problem, but as a regression problem. Öztürk *et al.* [2018] also learns molecular features and protein features in an automated manner using a CNN, unlike previous literature. They apply a CNN on the SMILE representation of the molecule and and the 2-D protein sequence, to learn the feature vector in an automated manner.

Ragoza *et al.* [2016] propose to leverage the 3-D nature of proteins and ligands by using a 3-D CNN.

Fout *et al.* [2017] propose the use of GCNs to identify the interface at which interaction between two proteins occurs. They represent a protein as a graph, where constituent amino-acids form the nodes, and the k-closest amino-acids are neighbours. Using this representation, they apply a GCN to learn a feature vector for each of the proteins. To find the interface between the two proteins, they classify whether any pair of amino-acid residues interact with each other.

Wu *et al.* [2019*a*] propose a unified RNN-CNN pipeline, which uses an RNN to learn the feature vector for proteins and ligands in an unsupervised manner, and the CNN to predict the binding affinity between the protein and the ligand in a supervised manner. They compare between a fingerprint representation of molecules [Wang *et al.*, 2009] and the SMILE representation. They also compare between various methods of representation of proteins, such as pfam domains [Wang *et al.*, 2009] and 3-D structure.

Wang and Zeng [2013] propose the use of a restricted Boltzman Machine [Rumelhart *et al.*, 1986] to predict drug target binding affinity and mode of action of binding. An RBM is a two-layer graphical model that can be used to learn a probability distribution over input data.

Feng *et al.* [2018] propose PADME (name inspired from the original trilogy), which is a deep learning framework for predicting drug target binding affinity. They learn ligand features using GCNs, and protein features using protein sequence composition. They feed these features into a DNN to predict the binding affinity between the protein and the ligand.

Ozturk *et al.* [2019] propose wideDTA, which uses textual representation of proteins and ligands to predict the binding affinity between them. WideDTA uses four text-based information sources, namely the protein sequence, ligand SMILES, protein domains and motifs, and maximum common substructure words to predict binding affinity.

There is still need for more powerful and accurate methods for predicting ligand protein binding affinity. However, before further technical advancements can be made, there is a need for standardisation of datasets and evaluation metrics. We need an analysis of the various protein and ligand representations. Further, there is also a need for comparison between automatically learning protein and ligand features against using a pre-defined set of features.

CHAPTER 5

Case Study: Using Experimental Measures for PPI Prediction

In the previous chapter, we had a brief look at the various computational methods which are generally used to predict protein-protein interactions. Most approaches are either experimental in nature or computational. Common experimental approaches are yeast to hybrid screen [Terentiev *et al.*, 2009] and mass spectroscopy. However, experimental approaches are expensive and noisy [Rajagopala *et al.*, 2012]. Computational methods rely on the hypothesis that two proteins which share a link are likely to share a common function or have evolved together. There are also machine learning methods which rely on either the sequence information of proteins or the 3-D structure of the protein.

While such approaches have been successful to some extent, we believe, that there should be a more effective way to combine the information presented by various computational methods. We present a framework which allows to combine various metrics between protein pairs. Further, our method work even for incomplete set of scores between protein pairs.

The work presented in this chapter has been done in collaboration with Baker Lab in University of Washington. It is primarily contributed by Aakash Srinivasan(CS14B060), and is also presented in his B.Tech project report.

Before describing our approach to the link prediction problem, we present the original approach, which has not yet been published, taken by our collaborators at Baker lab.

5.1 Prior Approach

The original motivation of our collaborators behind taking this approach was to validate co-evolution as a predictor for protein protein interactions. Hence, they identify a set of protein pairs which score highly in terms of co-evolution, and check the overlap with a gold standard of known protein pair interactions. They essentially filter out pairs of proteins at every stage which score below a certain threshold. To do this they use three scores, namely mutual information (MI), Gremlin and docking. This approach is applied on the *E. Coli* proteome which comprises 3583 proteins. For the *E. Coli* dataset, there exists a set of protein pairs which are known to interact with each other, based on prior studies. Hence we can say that this method is reliable if there is a significant overlap between the positive set we identify and the gold standard. They use MI as the initial filter as it is easy to compute, though less accurate compared to Gremlin and docking. The initial MI filter is followed by a filter based on Gremlin scores, which is less accurate compared to docking, though easier to compute. This is then followed by a Docking filter to obtain a set of protein pair interactions. Docking is the most accurate metric for predicting interactions between protein pairs among the three. Below we describe the data that we have.

5.1.1 Mutual Information

Mutual information (MI) is an inexpensive metric which indicates co-evolution between a pair of proteins. Since MI is inexpensive, it is computed for 5.2 million pairs. The scores were pre-processed and smoothed using Average Product Correlation (APC) [Dunn *et al.*, 2007]. From this set of scores, the top 50k are considered to be likely pairs which may interact with each other. Further, we add 10k pairs which belong to a noisy experimental positive PPI set.

5.1.2 Gremlin [Kamisetty *et al.*, 2013]

Gremlin (generative regularized models of proteins) is defined as a statistical model that captures co-evolution from multiple sequences. It does so by optimizing a pseudo likelihood objective. One of the problem with MI is that they are very noisy. Gremlin scores are less noisy, but very expensive and hence we do not have scores for all pairs. We are given two variations of Gremlin scores - Gremlin 1 and Gremlin 2. For Gremlin 1, we have scores for 22k pairs and for Gremlin 2, we have 35k pairs. These scores are correlated with a pearson correlation of 0.301.

5.1.3 Docking

Docking is an expensive but accurate co-evolution prediction method that uses 3D model of proteins to come up with the score. They are extremely expensive and are available for only 6k pairs. We are given 4 variations of Docking scores - Docking1, Docking2, Docking3, Docking4. These variations are correlated with a pearson correlation of around 0.7.

5.1.4 Positive Control

Positive control is a set of protein pairs which we know with high confidence interact with each other. This set consists of 3.6k pairs.

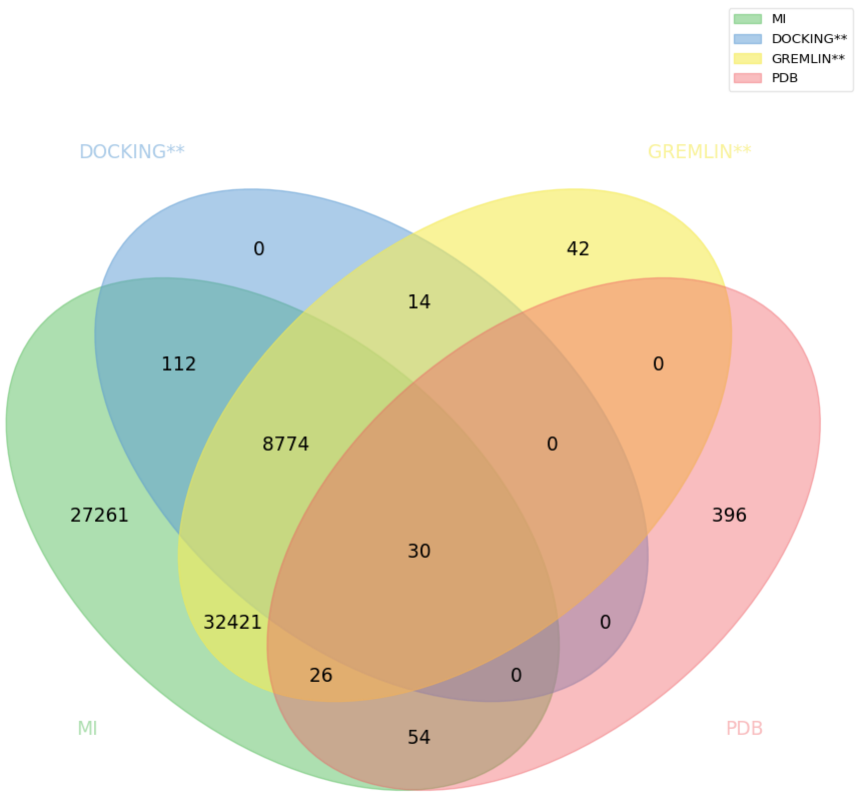

Figure 5.1: Intersection Analysis of various metrics

5.1.5 Negative Control

Negative control set consists of 283k protein pairs, which we know with high confidence do not interact with each other. 5.1 shows that all variations of Docking are well correlated with each other. Secondly, we find almost no correlation between MI and any other metric. Thirdly, we also find high amount of correlation between Gremlin and Docking scores, though we do not see a high amount of correlation among the Gremlin scores. In Figure 5.1 we see the intersections across

	MI	GREMLIN1	GREMLIN2	DOCKING1	DOCKING2	DOCKING3	DOCKING4
MI	1.	-0.015	0.023	0.052	0.04	0.066	0.046
GREMLIN1	-0.015	1.	0.301	0.45	0.595	0.485	0.498
GREMLIN2	0.023	0.301	1.	0.524	0.536	0.6	0.577
DOCKING1	0.052	0.45	0.524	1.	0.905	0.776	0.767
DOCKING2	0.04	0.595	0.536	0.905	1.	0.76	0.796
DOCKING3	0.066	0.485	0.6	0.776	0.76	1.	0.873
DOCKING4	0.046	0.498	0.577	0.767	0.796	0.873	1.

Table 5.1: Correlation matrix of various scores

MI, GREMLIN, Docking and PDB (positive control). This figure essentially summarises the filter out process followed by the approach taken by our collaborators at Baker Lab. Finally, 738 protein pairs are identified based on highest Docking scores. The overlap with the positive control set is 66 pairs, which is higher than the baseline (mass spectrometry), despite the predicted set having a much smaller size in this case. Using this result, we can conclude that co-evolution can act as a good predictor for PPI.

5.2 Our Approach

The approach described above helps us identify some novel protein interactions. However, it becomes difficult to identify an interaction if the interaction gets filtered out at the first step, as it has a low MI score. In such a case, the local network of MI scores around that protein pair might be able to help us classify the protein pair correctly. Further, the approach described above gives us no information about the likelihood of interaction between any of the filtered out pairs. We propose to build a multi-graph with the nodes as proteins, and weighted edges containing information about the various scoring metrics. We experiment with both unattributed nodes and using gene ontology (GO terms) [Ashburner *et al.*, 2000] information as node attributes. GO terms are indicative of the functionality of a particular protein. On this network, we wish to learn a link prediction algorithm. Our approach is summarised in Figure 5.2. Our link prediction algorithm relies on various heuristics between node pairs. We train a classifier on this set of heuristics to predict the likelihood of interaction between a pair of proteins. We experiment with both weighted and un-weighted graphs. To construct the un-weighted graph, we consider a link between a pair of nodes if the metric exists for a pair of nodes.

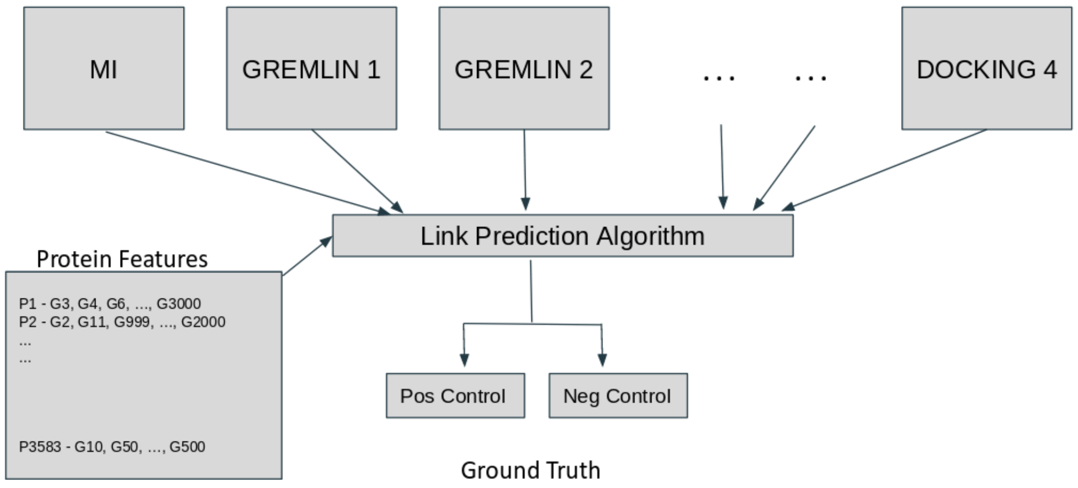
MI
GREMLIN 1
GREMLIN 2
. . .
. . .
DOCKING 4
Link Prediction Algorithm
Protein Features
P1 - G3, G4, G6, ..., G3000
P2 - G2, G11, G999, ..., G2000
...
...
P3583 - G10, G50, ..., G500
Pos Control
Neg Control
Ground Truth

Figure 5.2: Proposed pipeline for the PPI prediction problem

This becomes especially useful for noisy networks such as the filtered MI network, where we consider the top 50k MI scores along with 10k experimental pairs. The heuristics for both weighted and un-weighted graphs are summarised in Table 5.2. For a more detailed treatment of these heuristics, the reader is referred to [Zhu and Xia, 2016; Lichtenwalter *et al.*, 2010]. In the above table, nodes are denoted by u and v. $N(u)$ denotes the set of neighbours of node u. e_{ij} denoted the wight of the edge between nodes i and j. Propflow for a pair (u, v) is defined as the probability that the random walk starts at u, ends at v in l steps or fewer.

Each of these heuristics tries to capture information regarding the similarity of the neighbourhoods of a pair of nodes. Before we present our results in the next section, we discuss the evaluation metric used:

5.2.1 Evaluation Metric

We are interested in seeing how accurate our most confident positive predictions are. We focus on the area under precision recall curve within $\leq 20\%$ recall. Similarly, we are interested in True Positive Rate (TPR) within 0.2% False Positive Rate (FPR).

Heuristic	Un-Weighted	Weighted
Common Neighbours (CN)	$\|N(u) \cap N(v)\|$	$\sum_{x \in N(u) \cap N(v)} (w_{ux} + w_{vx})$
Adamic Adar (AA)	$\sum\limits_{x \in N(u) \cap N(v)} \frac{1}{log\|N(x)\|}$	$\sum_{x \in N(u) \cap N(v)} \frac{e_{ux}+e_{vx}}{log(\sum_{l \in N(x)} e_{xl})}$
Resource Allocation (RA)	$\sum\limits_{x \in N(u) \cap N(v)} \frac{1}{\|N(x)\|}$	$\sum_{x \in N(u) \cap N(v)} \frac{e_{ux}+e_{vx}}{\sum_{l \in N(x)} e_{xl}}$
Jaccard Index (JA)	$\frac{\|N(u) \cap N(v)\|}{\|N(u) \cup N(v)\|}$	$\frac{\sum\limits_{x \in N(u) \cap N(v)} (w_{ux}+w_{vx})}{\sum\limits_{x \in N(u)} w_{x,u} + \sum\limits_{x \in N(v)} w_{x,v}}$
Preferential Attachment (PA)	$\|N(u)\|.\|N(v)\|$	$(\sum_{x \in N(u)} e_{ux})(\sum_{x \in N(v)} e_{vx})$
Propflow	$\sum_{p \in Path(u,v), \|\|p\|\| < l} \prod_{p_k \in p} \frac{1}{wt(p_k)}$	$\sum_{p \in Path(u,v), \|\|p\|\| < l} \prod_{p_k \in p} \frac{e_{p_k p_{k+1}}}{wt(p_k)}$

Table 5.2: Network heuristics for link prediction

This means that we want to know how many positive controls are recovered within 0.002* 1.8 million = 3600 negative control pairs wrongly predicted as positive. We normalize the area under the curves by the maximum possible area (i.e max FPR $\leq$ 0.2%) and report these normalized scores. Note that the threshold of 20% is chosen somewhat arbitrarily.

5.3 Results

5.3.1 Using Plain Scores

In this setup, we try to predict interaction between a pair of proteins using just the scores we have(MI, Gremlin and Docking). This does not involve using any network information or learning. It is very similar to the approach described in section 5.1. We essentially place a threshold value on the score and classify everything above that threshold to be a positive link. These results give us an indicator of how reliable the scores are. Note that, this method is not generalisable, as it cannot make predictions for a pair of proteins for which we do not have a given score. For example, predictions for docking can be made only on the 5k pairs for which we have docking scores. In Table 5.3, (MI (with expt pairs)) refers to the union of top 50k MI scores and 10k pairs of proteins which are more likely to interact from previous experiments. The columns (Pos Control) and (Neg Control) contain the overlap between the score and the positive and negative control sets respectively. From the results, we see that Gremlin and Docking scores are fairly reliable. We also see that MI scores by themselves are an unreliable predictor. The only advantage of MI is that it is easy to compute and we have MI scores for a large number of protein pairs.

Score	Pos Control	Neg Control	AUROC <(0.002 FPR)	AUPR <(0.2 - recall)
MI (with expt pairs)	**1373**	16969	0.00134	0.03537
GREMLIN1	631	5642	0.23965	1
GREMLIN2	847	8953	0.24894	1
DOCKING1	337	1232	-	1
DOCKING2	330	1120	0.41212	1
DOCKING3	370	767	-	1
DOCKING4	364	813	0.43407	1
MI top 50000	**261**	16824	0.00623	0.11806
MI - all	911	808524	0.04612	0.05548

Table 5.3: Baseline models - Results

5.3.2 Using Weighted and Un-weighted Networks

Next we wish to use the network information presented to us by various metrics. We construct multiple graphs. Each graph contains the set of proteins as nodes, and a particular metric as the set of edges. We consider a weighted and an un-weighted variation of each of the graphs.

We have 8 networks, 4 corresponding to Docking, 2 for Gremlin and 2 for MI. We first consider all scores we have for MI. The second network built from MI considers only the top 50k pairs based on the MI scores. To this we add MI scores for 10k additional pairs, which are chosen from a noisy experimental setup.

For a particular graph, we compute the heuristics described in 5.2. Using these heuristics as input features, we try to learn a classifier. For training the classifier, we consider 1800 positive pairs, and 1.8 million negative control pairs. We use the remaining 1800 positive control pairs and 1.8 million negative control pairs for testing purposes. We choose the ratio between positive and negative control pairs to be 1000, as a randomly selected pair of proteins is 1000 times more likely to not interact than it is to interact with each other.

Scores	Weighted?	AUROC <(0.002 FPR)	AUPR <(0.2 - recall)
MI - all	Yes	**0.08243**	**0.13442**
	No	0.03673	0.04897
MI - exp	Yes	**0.15364**	**0.3246**
	No	**0.20874**	**0.48646**
Gremlin1	Yes	0.11377	0.22739
	No	0.09295	0.12262
Gremlin2	Yes	0.13512	0.21523
	No	0.13977	0.25433
Docking1	Yes	0.07827	0.12015
	No	0.07785	0.10329
Docking2	Yes	0.07506	0.11766
	No	0.07675	0.10644
Docking3	Yes	0.08856	0.15055
	No	0.09534	0.15986
Docking4	Yes	0.09622	0.17647
	No	0.09632	0.15691

Table 5.4: Results - Link Prediction Heuristics

There are a lot of interesting insights we get about our data from these results. Firstly, we find that the best performing dataset is using the top MI scores along with experimental data on an un-weighted graph. We believe that in this case the un-weighted graph performs better than the weighted graph simply because the weights of the pairs corresponding to the experimental set is very noisy. This is because these pairs do not have high MI scores, even though they are highly likely to interact. Considering an un-weighted graph helps us get rid of this noise.

We find that while using all MI scores the weighted graph performs better. Hence, we can say that having MI scores does help us in this task. We also find that we are able to beat the prediction of links using MI scores without network information (5.3). Hence, we can say that network information can help us predict network information in a better manner.

We find that Gremlin scores perform better than docking scores. This is probably because the size of the network formed by docking scores is small, and it becomes harder to learn due to paucity of data.

5.3.3 Watch Your Step

Next we attempt to use the watch your step(WYS) algorithm for the link prediction algorithm [Abu-El-Haija *et al.*, 2017]. Watch Your step is a state of the art algorithm for link prediction, which optimises random walk hyperparameters in an automated fashion by converting them into differentiable parameters. We try using the WYS algorithm for the complete set of MI scores and see an improvement in performance.

Scores	Weighted?	AUROC <(0.002 FPR)	AUPR <(0.2 - recall)
5 Hop - MI all	Yes	0.1284	0.22449
	No	0.01612	0.02389
3 Hop - MI all	Yes	0.13642	0.32607
	No	0.01914	0.0092
1 Hop - MI all	Yes	**0.14983**	**0.35335**
	No	**0.04203**	**0.03453**

Table 5.5: Results - Watch Your Step - Weighted and Un-weighted

Hence, we can say that WYS is able to learn network information in a better manner than the classifier learned on top of network heuristics. Again,we see a better performance for weighted graphs against un-weighted graphs. We also find a decrease in performance with an increase in number of hops for random walks. This is probably because of the lack of transitivity for MI.

5.3.4 GO Terms

Next, we try to use information from GO terms. These terms indicate the function performed by a protein. Hence, for each protein, we know a set of terms indicating whether the protein contributes to a particular function. Hence, prediction using GO terms relies on the hypothesis that a pair of proteins is more likely to interact if they share a common function.

Our approach for prediction using GO terms is summarised in Figure 5.3. For each pair of proteins, we combine the features using a hadamard product, and

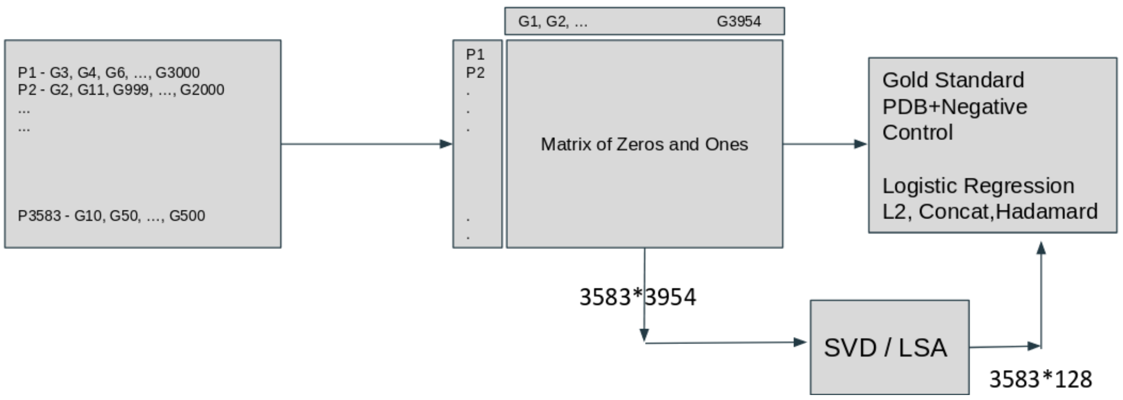

Figure 5.3: Algorithm for prediction using GO terms

train a classifier on this feature vector. We also experimented with concatenation of features, but obtained better results using a hadamard product. We find that, using just GO terms, we are able to obtain a significant jump in performance as compared to the MI network.

Scores	AUROC <(0.002 FPR)	AUPR <(0.2 - recall)
GO terms	0.34746	0.99119

Table 5.6: Results - GO Terms for link prediction

Since GO-terms are able to predict links better than any of our co-evolution metrics, we believe there is a need to include this information in our models. We believe that combining information from MI network and GO terms should be able to beat GO term predictions by a significant margin.

5.3.5 Positive Control Network

Here, we pose a very important baseline which our models also need to beat. In this setup we consider an un-weighted network consisting of 1800 links, taken from the positive control set. On this network, we attempt to train a classifier for link prediction using heuristics and WYS. We find that WYS on this setup outperforms all previous setups. Since, we are able to learn from such a small graph, we can say that the network is highly modular in nature, as is characteristic of PPI networks. t the moment, we are unable to beat this baseline. This essentially means that though network information is significant, the various co-evolution scores such as MI are not very helpful. We are however skeptical about making this conclusion as there are a lot of things we have not been able to try yet, which we describe briefly in the next section.

Method	AUROC <(0.002 FPR)	AUPR <(0.2 - recall)
Heuristics	0.32321	0.9414
WYS	0.35883	0.57994

Table 5.7: Results - Network from Positive Control

5.4 Conclusions and Future Work

We proposed a novel framework for PPI prediction which allows us to use similarity metrics between protein pairs. Our method allows us to predict for any arbitrary pair of proteins, from an incomplete network of metrics. Further, we can also combine the various metrics by building a multi-graph, and training our classifier on a combination of heuristics from each layer of the multi-graph.

We also propose two baseline models to highlight the utility of our approach. The first baseline we propose is to use just the metric scores, omitting any network information. Comparison to this baseline will help us understand how useful network information is. We have already beaten this baseline by a significant margin, and it is safe to say that network information can help us predict protein interactions more accurately. The second baseline we propose is to use just the positive links as a network, and attempt to predict the remaining links. Comparison to this approach will help us understand whether our model is able to learn from various metrics, namely, MI, Gremlin and Docking. Our current approach is unable to beat this baseline. However, there are plenty of experiments which may help us beat this baseline.

Firstly, we need to run the heuristics and WYS based approaches for the complete multi-graph of metrics, comprising MI, Gremlin and Docking. We can also add another layer to the multi-graph, comprising information of GO terms. However, information regarding GO terms maybe more effectively captured with the help of a GCN. Further, we believe that adding attention over edge weights in the GCN can also give significant as the edge metrics we have our very noisy.

We have clearly defined the framework and objective of this problem. However, due to paucity of time we were unable to attempt all of these experiments, and leave them for future work.

CHAPTER 6

Conclusion and Directions for Future Work

In this thesis, we looked at the various applications of deep learning to network biology and network chemistry. We also provided state of the art results for chemical reaction prediction on the USPTO dataset. Further, we also looked at the problem of predicting interactions between proteins from noisy and incomplete co-evolution metrics between protein pairs. We also looked at the shortcomings of each of our methods, and possible methods to overcome those. Current research helps us understand the extent to which applying deep learning to network data can be beneficial in reducing the cost of healthcare and material design. In this chapter, we wish to take a look at the broad direction of research for the application of deep learning methods in drug design.

The cost of drugs is often high not because of manufacturing costs, but rather because of developmental costs. Companies often spend 10-15 years and 600-900 million dollars on designing one drug. Further, many trials never see the light of day, and the loss incurred by this research needs to be recovered from the sale of successfully developed drugs. We believe that deep learning methods can help us speed up and reduce the cost of drug design significantly. For example, molecule generation can help us design molecules with some specific desirable properties. This is very useful for the problem of drug design, where efficiently traversing through the space of molecules is a significant challenge. Another example where deep learning can help optimise cost for drug design, is using a retro-synthesis prediction algorithm for finding the most efficient pathway for prediction of a molecule. Identifying novel protein interactions helps us improve our knowledge of the functioning of the body. All these methods contribute to our ability to design drugs at a faster pace and at a cheaper cost.

We understand that much of the academic research done currently is restricted to specific datasets, and may not be easily deployable in the field. To address this issue, we believe that industry-academia collaborations are of utmost importance. Another issue with academic research is that it deals with many small components of the drug design process, and there needs to be research which tries to bring together at least some of these works into one pipeline for designing new drugs. While the task of animal and human testing is irreplaceable, many of the other processes in drug design can be automated. We hope to see deep learning making a real difference in the field of drug design in the near future and not remain an exciting direction for future work.

REFERENCES

Abadi, M., A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and **X. Zheng** (2015). TensorFlow: Large-scale machine learning on heterogeneous systems. URL `http://tensorflow.org/`. Software available from tensorflow.org.

Abu-El-Haija, S., B. Perozzi, R. Al-Rfou, and **A. Alemi** (2017). Watch your step: Learning graph embeddings through attention. *CoRR*, **abs/1710.09599**. URL `http://arxiv.org/abs/1710.09599`.

Ashburner, M., C. A. Ball, J. A. Blake, D. Botstein, H. Butler, J. M. Cherry, A. P. Davis, K. Dolinski, S. S. Dwight, J. T. Eppig, *et al.* (2000). Gene ontology: tool for the unification of biology. *Nature genetics*, **25**(1), 25.

Bahdanau, D., K. Cho, and **Y. Bengio** (2015). Neural machine translation by jointly learning to align and translate. *CoRR*, **abs/1409.0473**.

Baylon, J. L., N. A. Cilfone, J. R. Gulcher, and **T. W. Chittenden** (2019). Enhancing retrosynthetic reaction prediction with deep learning using multiscale reaction classification. *Journal of Chemical Information and Modeling*, **59**(2), 673–688. URL `https://doi.org/10.1021/acs.jcim.8b00801`.

Benhenda, M. (2017). Chemgan challenge for drug discovery: can ai reproduce natural chemical diversity?

Bhagat, S., G. Cormode, and **S. Muthukrishnan** (2011). Node classification in social networks. *CoRR*, **abs/1101.3291**. URL `http://arxiv.org/abs/1101.3291`.

Blaschke, T., M. Olivecrona, O. Engkvist, J. Bajorath, and **H. Chen** (2017). Application of generative autoencoder in de novo molecular design. *CoRR*, **abs/1711.07839**. URL `http://arxiv.org/abs/1711.07839`.

Bradshaw, J., M. J. Kusner, B. Paige, M. H. S. Segler, and **J. M. Hernndez-Lobato**, A generative model for electron paths. *In International Conference on Learning Representations*. 2019. URL `https://openreview.net/forum?id=r1x4BnCqKX`.

Camacho, D. M., K. M. Collins, R. K. Powers, J. C. Costello, and **J. J. Collins** (2018). Next-generation machine learning for biological networks. *Cell*, **173**, 1581–1592.

Cao, N. D. and **T. Kipf** (2018). Molgan: An implicit generative model for small molecular graphs.

Chung, J., **Ç. Gülçehre**, **K. Cho**, and **Y. Bengio** (2014). Empirical evaluation of gated recurrent neural networks on sequence modeling. *CoRR*, **abs/1412.3555**. URL http://arxiv.org/abs/1412.3555.

Coley, C., **W. Jin**, **L. Rogers**, **T. F. Jamison**, **T. S. Jaakkola**, **W. H. Green**, **R. Barzilay**, and **K. F. Jensen** (2019). A graph-convolutional neural network model for the prediction of chemical reactivity. *Chem. Sci.*, **10**, 370–377. URL http://dx.doi.org/10.1039/C8SC04228D.

Coley, C. W., **R. Barzilay**, **T. S. Jaakkola**, **W. H. Green**, and **K. F. Jensen** (2017). Prediction of organic reaction outcomes using machine learning. *ACS Central Science*, **3**(5), 434–443. URL https://doi.org/10.1021/acscentsci.7b00064.

Corey, E. J. (1971). Centenary lecture. computer-assisted analysis of complex synthetic problems. *Quarterly Reviews, Chemical Society*, **25**(4), 455. URL https://doi.org/10.1039/qr9712500455.

Dahl, G. E., **N. Jaitly**, and **R. R. Salakhutdinov** (2014). Multi-task neural networks for qsar predictions. *CoRR*, **abs/1406.1231**.

Dai, H., **Y. Tian**, **B. Dai**, **S. Skiena**, and **L. Song** (2018). Syntax-directed variational autoencoder for structured data. *CoRR*, **abs/1802.08786**. URL http://arxiv.org/abs/1802.08786.

Dale, J. M., **L. Popescu**, and **P. D. Karp** (2010). Machine learning methods for metabolic pathway prediction. *BMC Bioinformatics*, **11**(1), 15. URL https://doi.org/10.1186/1471-2105-11-15.

Do, K., **T. Tran**, and **S. Venkatesh** (2019). GRAPH TRANSFORMATION POLICY NETWORK FOR CHEMICAL REACTION PREDICTION. URL https://openreview.net/forum?id=r1f78iAcFm.

Douglas, B. L. (2011). The weisfeiler-lehman method and graph isomorphism testing.

Dunn, S. D., **L. M. Wahl**, and **G. B. Gloor** (2007). Mutual information without the influence of phylogeny or entropy dramatically improves residue contact prediction. *Bioinformatics*, **24**(3), 333–340.

Elton, D. C., **Z. Boukouvalas**, **M. D. Fuge**, and **P. W. Chung** (2019). Deep learning for molecular generation and optimization - a review of the state of the art. *CoRR*, **abs/1903.04388**. URL http://arxiv.org/abs/1903.04388.

Engkvist, O., **P.-O. Norrby**, **N. Selmi**, **Y. hong Lam**, **Z. Peng**, **E. C. Sherer**, **W. Amberg**, **T. Erhard**, and **L. A. Smyth** (2018). Computational prediction of chemical reactions: current status and outlook. *Drug Discovery Today*, **23**(6), 1203–1218. URL https://doi.org/10.1016/j.drudis.2018.02.014.

Fagerberg, R., **C. Flamm**, **R. Kianian**, **D. Merkle**, and **P. F. Stadler** (2018). Finding the k best synthesis plans. *Journal of Cheminformatics*, **10**(1). URL https://doi.org/10.1186/s13321-018-0273-z.

Feng, Q., **E. V. Dueva**, **A. Cherkasov**, and **M. Ester** (2018). PADME: A deep learning-based framework for drug-target interaction prediction. *CoRR*, **abs/1807.09741**. URL http://arxiv.org/abs/1807.09741.

Fionda, V., *Networks in Biology*. 2018. ISBN 9780128096338.

Fooshee, D., **A. Mood**, **E. Gutman**, **M. Tavakoli**, **G. Urban**, **F. Liu**, **N. Huynh**, **D. Van Vranken**, and **P. Baldi** (2018). Deep learning for chemical reaction prediction. *Mol. Syst. Des. Eng.*, **3**, 442–452. URL http://dx.doi.org/10.1039/C7ME00107J.

Fout, A., **J. Byrd**, **B. Shariat**, and **A. Ben-Hur**, Protein interface prediction using graph convolutional networks. *In NIPS*. 2017.

Friedman, N., **M. Linial**, **I. Nachman**, and **D. Pe'er**, Using bayesian networks to analyze expression data. *In Proceedings of the Fourth Annual International Conference on Computational Molecular Biology*, RECOMB '00. ACM, New York, NY, USA, 2000. ISBN 1-58113-186-0. URL http://doi.acm.org/10.1145/332306.332355.

Gasteiger, J., **M. G. Hutchings**, **B. Christoph**, **L. Gann**, **C. Hiller**, **P. Löw**, **M. Marsili**, **H. Saller**, and **K. Yuki**, A new treatment of chemical reactivity: Development of EROS, an expert system for reaction prediction and synthesis design. *In Topics in Current Chemistry*. Springer Berlin Heidelberg, 1987, 19–73. URL https://doi.org/10.1007/3-540-16904-0_14.

Gaulton, A., **A. Hersey**, **M. Nowotka**, **A. P. Bento**, **J. Chambers**, **D. Mendez**, **P. Mutowo**, **F. Atkinson**, **L. J. Bellis**, **E. Cibrián-Uhalte**, **M. Davies**, **N. Dedman**, **A. Karlsson**, **M. P. Magariños**, **J. P. Overington**, **G. Papadatos**, **I. Smit**, and **A. R. Leach** (2016). The ChEMBL database in 2017. *Nucleic Acids Research*, **45**(D1), D945–D954. URL https://doi.org/10.1093/nar/gkw1074.

Gelernter, H., **J. R. Rose**, and **C. Chen** (1990). Building and refining a knowledge base for synthetic organic chemistry via the methodology of inductive and deductive machine learning. *Journal of Chemical Information and Computer Sciences*, **30**(4), 492–504. ISSN 0095-2338. URL https://pubs.acs.org/doi/abs/10.1021/ci00068a023.

Goh, G. B., **N. O. Hodas**, and **A. Vishnu** (2017*a*). Deep learning for computational chemistry.

Goh, G. B., **C. Siegel**, **A. Vishnu**, **N. O. Hodas**, and **N. Baker** (2017*b*). Chemception: A deep neural network with minimal chemistry knowledge matches the performance of expert-developed qsar/qspr models.

Gómez-Bombarelli, R., **J. N. Wei**, **D. Duvenaud**, **J. M. Hernández-Lobato**, **B. Sánchez-Lengeling**, **D. Sheberla**, **J. Aguilera-Iparraguirre**, **T. D. Hirzel**, **R. P. Adams**, and **A. Aspuru-Guzik** (2018). Automatic chemical design using a data-driven continuous representation of molecules. *ACS Central Science*, **4**(2), 268–276. URL https://doi.org/10.1021/acscentsci.7b00572.

Goodfellow, I., **J. Pouget-Abadie**, **M. Mirza**, **B. Xu**, **D. Warde-Farley**, **S. Ozair**, **A. Courville**, and **Y. Bengio**, Generative adversarial nets. *In* **Z. Ghahramani**, **M. Welling**, **C. Cortes**, **N. D. Lawrence**, and **K. Q. Weinberger** (eds.), *Advances in Neural Information Processing Systems 27*. Curran Associates, Inc., 2014, 2672–2680. URL http://papers.nips.cc/paper/5423-generative-adversarial-nets.pdf.

Grover, A. and **J. Leskovec** (2016). node2vec: Scalable feature learning for networks. *CoRR*, **abs/1607.00653**. URL http://arxiv.org/abs/1607.00653.

Grzybowski, B. A., **S. Szymkuć**, **E. P. Gajewska**, **K. Molga**, **P. Dittwald**, **A. Wołos**, and **T. Klucznik** (2018). Chematica: A story of computer code that started to think like a chemist. *Chem*, **4**(3), 390–398. URL `https://doi.org/10.1016/j.chempr.2018.02.024`.

Guimaraes, G. L., **B. Sanchez-Lengeling**, **P. L. C. Farias**, and **A. Aspuru-Guzik** (2017). Objective-reinforced generative adversarial networks (organ) for sequence generation models. *CoRR*, **abs/1705.10843**.

Guimerà, R. and **L. A. N. Amaral** (2005). Functional cartography of complex metabolic networks. *Nature*, **433**, 895–900.

Hamilton, W. L., **R. Ying**, and **J. Leskovec** (2017*a*). Inductive representation learning on large graphs. *CoRR*, **abs/1706.02216**. URL `http://arxiv.org/abs/1706.02216`.

Hamilton, W. L., **R. Ying**, and **J. Leskovec** (2017*b*). Representation learning on graphs: Methods and applications. *CoRR*, **abs/1709.05584**. URL `http://arxiv.org/abs/1709.05584`.

Henderson, K., **B. Gallagher**, **T. Eliassi-Rad**, **H. Tong**, **S. Basu**, **L. Akoglu**, **D. Koutra**, **C. Faloutsos**, and **L. Li**, Rolx: Structural role extraction &#38; mining in large graphs. *In Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '12. ACM, New York, NY, USA, 2012. ISBN 978-1-4503-1462-6. URL `http://doi.acm.org/10.1145/2339530.2339723`.

Herges, R. (1994). Coarctate transition states: the discovery of a reaction principle. *Journal of Chemical Information and Computer Sciences*, **34**, 91–102.

Hochreiter, S. and **J. Schmidhuber** (1997). Long short-term memory. *Neural Computation*, **9**(8), 1735–1780. URL `https://doi.org/10.1162/neco.1997.9.8.1735`.

Hulovatyy, Y., **R. W. Solava**, and **T. Milenković** (2014). Revealing missing parts of the interactome via link prediction. *PLoS ONE*, **9**(3), e90073. URL `https://doi.org/10.1371/journal.pone.0090073`.

Irwin, J. J. and **B. K. Shoichet** (2005). Zinc a free database of commercially available compounds for virtual screening. *Journal of Chemical Information and Modeling*, **45**(1), 177–182. URL `https://doi.org/10.1021/ci049714+`. PMID: 15667143.

Jaques, N., **S. Gu**, **R. E. Turner**, and **D. Eck** (2016). Tuning recurrent neural networks with reinforcement learning. *CoRR*, **abs/1611.02796**. URL `http://arxiv.org/abs/1611.02796`.

Jin, W., **R. Barzilay**, and **T. S. Jaakkola** (2018). Junction tree variational autoencoder for molecular graph generation. *CoRR*, **abs/1802.04364**. URL `http://arxiv.org/abs/1802.04364`.

Jin, W., **C. Coley**, **R. Barzilay**, and **T. Jaakkola**, Predicting organic reaction outcomes with weisfeiler-lehman network. *In* **I. Guyon**, **U. V. Luxburg**, **S. Bengio**, **H. Wallach**, **R. Fergus**, **S. Vishwanathan**, and **R. Garnett** (eds.), *Advances in Neural Information Processing Systems 30*. Curran Associates, Inc., 2017*a*, 2607–2616. URL `http://papers.nips.cc/paper/6854-predicting-organic-reaction-outcomes-with-weisfeiler-lehman-network.pdf`.

Jin, W., **C. W. Coley**, **R. Barzilay**, and **T. S. Jaakkola** (2017*b*). Predicting organic reaction outcomes with weisfeiler-lehman network. *CoRR*, **abs/1709.04555**. URL `http://arxiv.org/abs/1709.04555`.

Jorgensen, W. L., **E. R. Laird**, **A. J. Gushurst**, **J. M. Fleischer**, **S. A. Gothe**, **H. E. Helson**, **G. D. Paderes**, and **S. Sinclair** (1990). CAMEO: a program for the logical prediction of the products of organic reactions. *Pure and Applied Chemistry*, **62**(10), 1921–1932. URL `https://doi.org/10.1351/pac199062101921`.

Kamisetty, H., **S. Ovchinnikov**, and **D. Baker** (2013). Assessing the utility of coevolution-based residue–residue contact predictions in a sequence-and structure-rich era. *Proceedings of the National Academy of Sciences*, **110**(39), 15674–15679.

Karp, P. D., **A. Kothari**, **C. A. Fulcher**, **I. M. Keseler**, **M. Latendresse**, **M. Krummenacker**, **P. Subhraveti**, **Q. Ong**, **R. Billington**, **R. Caspi**, **S. M. Paley**, **W. K. Ong**, and **P. E. Midford** (2017). The BioCyc collection of microbial genomes and metabolic pathways. URL `https://doi.org/10.1093/bib/bbx085`.

Kayala, M. A. and **P. F. Baldi**, A machine learning approach to predict chemical reactions. *In* **J. Shawe-Taylor**, **R. S. Zemel**, **P. L. Bartlett**, **F. Pereira**, and **K. Q. Weinberger** (eds.), *Advances in Neural Information Processing Systems 24*. Curran Associates, Inc., 2011, 747–755. URL `http://papers.nips.cc/paper/4356-a-machine-learning-approach-to-predict-chemical-reactions.pdf`.

Keiser, M. J., **B. L. Roth**, **B. N. Armbruster**, **P. Ernsberger**, **J. J. Irwin**, and **B. K. Shoichet** (2007). Relating protein pharmacology by ligand chemistry. *Nature Biotechnology*, **25**(2), 197–206. URL `https://doi.org/10.1038/nbt1284`.

Khan, A. A., **B. Neyshabur**, **S. Hashemifar**, and **J. Xu** (2018). Predicting proteinprotein interactions through sequence-based deep learning. *Bioinformatics*, **34**(17), i802–i810. ISSN 1367-4803. URL `https://doi.org/10.1093/bioinformatics/bty573`.

Kim, Y., **B. Min**, and **G.-S. Yi** (2012). Iddi: Integrated domain-domain interaction and protein interaction analysis system. *Proteome science*, **10 Suppl 1**, S9.

Kingma, D. P. and **M. Welling** (2014). Auto-encoding variational bayes. *CoRR*, **abs/1312.6114**.

Kipf, T. N. and **M. Welling** (2016). Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*.

Kothari, A., **C. Fulcher**, **D. Weaver**, **I. M. Keseler**, **M. Latendresse**, **M. Krummenacker**, **P. Subhraveti**, **Q. Ong**, **R. Billington**, **R. Caspi**, **S. Paley**, **P. D. Karp**, **A. Mackie**, **I. Paulsen**, **A. Santos-Zavaleta**, **C. Bonavides-Martnez**, **D. A. Velzquez-Ramrez**, **J. Collado-Vides**, **L. Muiz-Rascado**, **M. Peralta-Gil**, and **S. Gama-Castro** (2016). The EcoCyc database: reflecting new knowledge about Escherichia coli K-12. *Nucleic Acids Research*, **45**(D1), D543–D550. ISSN 0305-1048. URL `https://doi.org/10.1093/nar/gkw1003`.

Lei, C. and **J. Ruan** (2012). A novel link prediction algorithm for reconstructing protein-protein interaction networks by topological similarity. *Bioinformatics*, **29**(3), 355–364. ISSN 1367-4803. URL `https://doi.org/10.1093/bioinformatics/bts688`.

Li, J., **D. Cai**, and **X. He** (2017). Learning graph-level representation for drug discovery. *CoRR*, **abs/1709.03741**. URL `http://arxiv.org/abs/1709.03741`.

Li, R., **S. Wang**, **F. Zhu**, and **J. Huang** (2018). Adaptive graph convolutional neural networks. *CoRR*, **abs/1801.03226**. URL `http://arxiv.org/abs/1801.03226`.

Lichtenwalter, R. N., **J. T. Lussier**, and **N. V. Chawla**, New perspectives and methods in link prediction. *In Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '10. ACM, New York, NY, USA, 2010. ISBN 978-1-4503-0055-1. URL `http://doi.acm.org/10.1145/1835804.1835837`.

Liu, B., **B. Ramsundar**, **P. Kawthekar**, **J. Shi**, **J. Gomes**, **Q. L. Nguyen**, **S. Ho**, **J. Sloane**, **P. Wender**, and **V. Pande** (2017). Retrosynthetic reaction prediction using neural sequence-to-sequence models. *ACS Central Science*, **3**(10), 1103–1113. URL `https://doi.org/10.1021/acscentsci.7b00303`.

Liu, Q., **M. Allamanis**, **M. Brockschmidt**, and **A. L. Gaunt** (2018*a*). Constrained graph variational autoencoders for molecule design. *CoRR*, **abs/1805.09076**. URL `http://arxiv.org/abs/1805.09076`.

Liu, Z., **C. Chen**, **L. Li**, **J. Zhou**, **X. Li**, and **L. Song** (2018*b*). Geniepath: Graph neural networks with adaptive receptive paths. *CoRR*, **abs/1802.00910**. URL `http://arxiv.org/abs/1802.00910`.

Lowe, D. (2017). Chemical reactions from US patents (1976-Sep2016). URL `https://figshare.com/articles/Chemical_reactions_from_US_patents_1976-Sep2016_/5104873`.

Lowe, D. M. (2012). Extraction of chemical structures and reactions from the literature. URL `https://www.repository.cam.ac.uk/handle/1810/244727`.

Luong, M., **E. Brevdo**, and **R. Zhao** (2017). Neural machine translation (seq2seq) tutorial. *https://github.com/tensorflow/nmt*.

Makhzani, A., **J. Shlens**, **N. Jaitly**, and **I. J. Goodfellow** (2015). Adversarial autoencoders. *CoRR*, **abs/1511.05644**. URL `http://arxiv.org/abs/1511.05644`.

Martino, A., **J.-L. Faulon**, **S. Martin**, and **Z. Zhang** (2007). Boolean dynamics of genetic regulatory networks inferred from microarray time series data. *Bioinformatics*, **23**(7), 866–874. ISSN 1367-4803. URL `https://doi.org/10.1093/bioinformatics/btm021`.

Mithani, A., **G. M. Preston**, and **J. Hein** (2009). Rahnuma: hypergraph-based tool for metabolic pathway prediction and network comparison. *Bioinformatics*, **25 14**, 1831–2.

Morgan, H. L. (1965). The generation of a unique machine description for chemical structures-a technique developed at chemical abstracts service. *Journal of Chemical Documentation*, **5**(2), 107–113. URL `https://doi.org/10.1021/c160017a018`.

Morishima, K., **M. Tanabe**, **M. Furumichi**, **M. Kanehisa**, and **Y. Sato** (2018). New approach for understanding genome variations in KEGG. *Nucleic Acids Research*, **47**(D1), D590–D595. ISSN 0305-1048. URL `https://doi.org/10.1093/nar/gky962`.

Nam, J. and **J. Kim** (2016). Linking the neural machine translation and the prediction of organic chemistry reactions. *CoRR*, **abs/1612.09529**. URL `http://arxiv.org/abs/1612.09529`.

Olivecrona, M., **T. Blaschke**, **O. Engkvist**, and **H. Chen** (2017). Molecular de novo design through deep reinforcement learning. *CoRR*, **abs/1704.07555**. URL `http://arxiv.org/abs/1704.07555`.

Öztürk, H., **A. Özgür**, and **E. Ozkirimli** (2018). DeepDTA: deep drug–target binding affinity prediction. *Bioinformatics*, **34**(17), i821–i829. URL `https://doi.org/10.1093/bioinformatics/bty593`.

Ozturk, H., **E. Ozkirimli**, and **A. Ozgur**, Widedta: prediction of drug-target binding affinity. 2019.

Păun, G. (2000). Computing with membranes. *Journal of Computer and System Sciences*, **61**(1), 108–143. URL `https://doi.org/10.1006/jcss.1999.1693`.

Peng, W., **J. Wang**, **J. Cai**, **L. Chen**, **M. Li**, and **F.-X. Wu** (2014). Improving protein function prediction using domain and protein complexes in ppi networks. *BMC systems biology*, **8**, 35.

Prado-Prado, F., **X. García-Mera**, **M. Escobar**, **E. Sobarzo-Sánchez**, **M. Yañez**, **P. Riera-Fernandez**, and **H. González-Díaz** (2011). 2d MI-DRAGON: A new predictor for protein–ligands interactions and theoretic-experimental studies of US FDA drug-target network, oxoisoaporphine inhibitors for MAO-a and human parasite proteins. *European Journal of Medicinal Chemistry*, **46**(12), 5838–5851. URL `https://doi.org/10.1016/j.ejmech.2011.09.045`.

Ragoza, M., **J. Hochuli**, **E. Idrobo**, **J. Sunseri**, and **D. R. Koes** (2016). Protein-ligand scoring with convolutional neural networks.

Ragoza, M., **J. Hochuli**, **E. Idrobo**, **J. Sunseri**, and **D. R. Koes** (2017). Protein–ligand scoring with convolutional neural networks. *Journal of Chemical Information and Modeling*, **57**(4), 942–957. URL `https://doi.org/10.1021/acs.jcim.6b00740`.

Rajagopala, S. V., **P. Sikorski**, **J. H. Caufield**, **A. Tovchigrechko**, and **P. Uetz** (2012). Studying protein complexes by the yeast two-hybrid system. *Methods*, **58**(4), 392–399. URL `https://doi.org/10.1016/j.ymeth.2012.07.015`.

Ramadan, E., **A. Tarafdar**, and **A. Pothen** (2004). A hypergraph model for the yeast protein complex network. *18th International Parallel and Distributed Processing Symposium, 2004. Proceedings.*, 189–.

Ramakrishnan, R., **P. O. Dral**, **M. Rupp**, and **O. A. von Lilienfeld** (2014). Quantum chemistry structures and properties of 134 kilo molecules. *Scientific Data*, **1**.

Raman, K. (2010). Construction and analysis of protein–protein interaction networks. *Automated Experimentation*, **2**(1), 2. URL `https://doi.org/10.1186/1759-4499-2-2`.

Rasmussen, C. E. and **C. K. I. Williams**, *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning series)*. The MIT Press, 2005. ISBN 9780262182539. URL `https://www.amazon.com/Gaussian-Processes-Learning-Adaptive-Computation/dp/026218253X?SubscriptionId=AKIAIOBINVZYXZQZ2U3A&tag=chimbori05-20&linkCode=xm2&camp=2025&creative=165953&creativeASIN=026218253X`.

Richoux, F., **C. Servantie**, **C. Borès**, and **S. Téletchéa** (2019). Comparing two deep learning sequence-based models for protein-protein interaction prediction. *CoRR*, **abs/1901.06268**. URL `http://arxiv.org/abs/1901.06268`.

Rogers, D. and **M. Hahn** (2010). Extended-connectivity fingerprints. *Journal of Chemical Information and Modeling*, **50**(5), 742–754. URL `https://doi.org/10.1021/ci100050t`.

Ruddigkeit, L., **R. van Deursen**, **L. C. Blum**, and **J.-L. Reymond** (2012). Enumeration of 166 billion organic small molecules in the chemical universe database GDB-17. *Journal of Chemical Information and Modeling*, **52**(11), 2864–2875. URL `https://doi.org/10.1021/ci300415d`.

Rumelhart, D. E., **J. L. McClelland**, and **C. PDP Research Group** (eds.), *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol. 1: Foundations*. MIT Press, Cambridge, MA, USA, 1986. ISBN 0-262-68053-X.

Samanta, B., **A. De**, **N. Ganguly**, and **M. Gomez-Rodriguez** (2018). Designing random graph models using variational autoencoders with applications to chemical design. *CoRR*, **abs/1802.05283**. URL `http://arxiv.org/abs/1802.05283`.

Sankar, A., **S. Ranu**, and **K. Raman** (2017). Predicting novel metabolic pathways through subgraph mining. *Bioinformatics*, **33**(24), 3955–3963. ISSN 1367-4803. URL `https://doi.org/10.1093/bioinformatics/btx481`.

Satoh, H. and **K. Funatsu** (1996). Further development of a reaction generator in the SOPHIA system for organic reaction prediction. knowledge-guided addition of suitable atoms and/or atomic groups to product skeleton. *Journal of Chemical Information and Computer Sciences*, **36**(2), 173–184. URL `https://doi.org/10.1021/ci950058a`.

Scarselli, F., **M. Gori**, **A. C. Tsoi**, **M. Hagenbuchner**, and **G. Monfardini** (2009). The graph neural network model. *Trans. Neur. Netw.*, **20**(1), 61–80. ISSN 1045-9227. URL `http://dx.doi.org/10.1109/TNN.2008.2005605`.

Schlichtkrull, M., **T. Kipf**, **P. Bloem**, **R. Berg**, **I. Titov**, and **M. Welling**, *Modeling Relational Data with Graph Convolutional Networks*. 2018, 593–607.

Schwaller, P., **T. Gaudin**, **D. Lányi**, **C. Bekas**, and **T. Laino** (2018). “found in translation”: predicting outcomes of complex organic chemistry reactions using neural sequence-to-sequence models. *Chemical Science*, **9**(28), 6091–6098. URL `https://doi.org/10.1039/c8sc02339e`.

Segler, M. H. S., **T. Kogej**, **C. Tyrchan**, and **M. P. Waller** (2017). Generating focussed molecule libraries for drug discovery with recurrent neural networks. *CoRR*, **abs/1701.01329**. URL `http://arxiv.org/abs/1701.01329`.

Segler, M. H. S., **M. Preuss**, and **M. P. Waller** (2018). Planning chemical syntheses with deep neural networks and symbolic AI. *Nature*, **555**(7698), 604–610. URL `https://doi.org/10.1038/nature25978`.

Simonovsky, M. and **N. Komodakis** (2018). Graphvae: Towards generation of small graphs using variational autoencoders. *CoRR*, **abs/1802.03480**. URL `http://arxiv.org/abs/1802.03480`.

Socorro, I. M., K. Taylor, and **J. M. Goodman** (2005). ROBIA: a reaction prediction program. *Organic Letters*, **7**(16), 3541–3544. URL `https://doi.org/10.1021/ol0512738`.

Szabo, A., G. Vattay, and **D. Kondor**, A neural nanonetwork model based on cell signaling molecules. 2011.

Terentiev, A. A., N. T. Moldogazieva, and **K. V. Shaitan** (2009). Dynamic proteomics in modeling of the living cell. protein-protein interactions. *Biochemistry (Moscow)*, **74**(13), 1586–1607. URL `https://doi.org/10.1134/s0006297909130112`.

Tian, K., M. Shao, Y. Wang, J. Guan, and **S. Zhou** (2016). Boosting compound-protein interaction prediction by deep learning. *Methods*, **110**, 64–72. URL `https://doi.org/10.1016/j.ymeth.2016.06.024`.

Veličković, P., G. Cucurull, A. Casanova, A. Romero, P. Liò, and **Y. Bengio** (2018). Graph Attention Networks. *International Conference on Learning Representations*. URL `https://openreview.net/forum?id=rJXMpikCZ`. Accepted as poster.

Vijayan, P., Y. Chandak, M. M. Khapra, and **B. Ravindran** (2018). Fusion graph convolutional networks. *CoRR*, **abs/1805.12528**. URL `http://arxiv.org/abs/1805.12528`.

Vilar, S., E. Quezada, E. Uriarte, S. Costanzi, F. Borges, D. Viña, and **G. Hripcsak** (2016). Computational drug target screening through protein interaction profiles. *Scientific Reports*, **6**(1). URL `https://doi.org/10.1038/srep36969`.

Virshup, A. M., J. Contreras-García, P. Wipf, W. Yang, and **D. N. Beratan** (2013). Stochastic voyages into uncharted chemical space produce a representative library of all possible drug-like compounds. *Journal of the American Chemical Society*, **135**(19), 7296–7303. URL `https://doi.org/10.1021/ja401184g`.

Vishwanathan, S. V. N., N. N. Schraudolph, R. Kondor, and **K. M. Borgwardt** (2010). Graph kernels. *J. Mach. Learn. Res.*, **11**, 1201–1242. ISSN 1532-4435. URL `http://dl.acm.org/citation.cfm?id=1756006.1859891`.

Voulodimos, A., N. Doulamis, A. Doulamis, and **E. Protopapadakis** (2018). Deep learning for computer vision: A brief review. *Computational Intelligence and Neuroscience*, **2018**, 1–13. URL `https://doi.org/10.1155/2018/7068349`.

Wang, Y., J. Xiao, T. O. Suzek, J. Zhang, J. Wang, and **S. H. Bryant** (2009). Pubchem: a public information system for analyzing bioactivities of small molecules. *Nucleic Acids Res*, 623–633.

Wang, Y. and **J. Zeng** (2013). Predicting drug-target interactions using restricted boltzmann machines. *Bioinformatics*, **29**(13), i126–i134. URL `https://doi.org/10.1093/bioinformatics/btt234`.

Wei, J. N., D. Duvenaud, and **A. Aspuru-Guzik** (2016). Neural networks for the prediction of organic chemistry reactions. *ACS Central Science*, **2**(10), 725–732. URL `https://doi.org/10.1021/acscentsci.6b00219`.

Wu, D., M. Karimi, Y. Shen, and **Z. Wang** (2019*a*). DeepAffinity: interpretable deep learning of compoundprotein affinity through unified recurrent and convolutional neural networks. URL `https://doi.org/10.1093/bioinformatics/btz111`.

Wu, Z., **S. Pan**, **F. Chen**, **G. Long**, **C. Zhang**, and **P. S. Yu** (2019*b*). A comprehensive survey on graph neural networks. *CoRR*, **abs/1901.00596**. URL http://arxiv.org/abs/1901.00596.

Wu, Z., **B. Ramsundar**, **E. N. Feinberg**, **J. Gomes**, **C. Geniesse**, **A. S. Pappu**, **K. Leswing**, and **V. S. Pande** (2017). Moleculenet: A benchmark for molecular machine learning. *CoRR*, **abs/1703.00564**. URL http://arxiv.org/abs/1703.00564.

Xu, Y., **K. Lin**, **S. Wang**, **L. Wang**, **C. Cai**, **C. Song**, **L. Lai**, and **J. Pei** (2019). Deep learning for molecular generation. *Future Medicinal Chemistry*, **11**(6), 567–597. URL https://doi.org/10.4155/fmc-2018-0358.

Yadati, N., **V. Nitin**, **M. Nimishakavi**, **P. Yadav**, **A. Louis**, and **P. Talukdar** (2019). Link prediction in hypergraphs using graph convolutional networks. URL https://openreview.net/forum?id=ryeaZhRqFm.

Yang, Y., **R. N. Lichtenwalter**, and **N. V. Chawla** (2015). Evaluating link prediction methods. *CoRR*, **abs/1505.04094**. URL http://arxiv.org/abs/1505.04094.

You, J., **B. Liu**, **R. Ying**, **V. S. Pande**, and **J. Leskovec** (2018*a*). Graph convolutional policy network for goal-directed molecular graph generation. *CoRR*, **abs/1806.02473**. URL http://arxiv.org/abs/1806.02473.

You, J., **R. Ying**, **X. Ren**, **W. L. Hamilton**, and **J. Leskovec** (2018*b*). Graphrnn: A deep generative model for graphs. *CoRR*, **abs/1802.08773**. URL http://arxiv.org/abs/1802.08773.

Yu, L., **W. Zhang**, **J. Wang**, and **Y. Yu** (2016). Seqgan: Sequence generative adversarial nets with policy gradient. *CoRR*, **abs/1609.05473**. URL http://arxiv.org/abs/1609.05473.

Zhang, M. and **Y. Chen** (2018). Link prediction based on graph neural networks. *CoRR*, **abs/1802.09691**. URL http://arxiv.org/abs/1802.09691.

Zhu, B. and **Y. Xia** (2016). Link prediction in weighted networks: A weighted mutual information model. *PLOS ONE*, **11**(2), e0148265. URL https://doi.org/10.1371/journal.pone.0148265.

Zitnik, M. and **J. Leskovec** (2017). Predicting multicellular function through multi-layer tissue networks. *Bioinformatics*, **33**(14), i190–i198. ISSN 1367-4803. URL https://doi.org/10.1093/bioinformatics/btx252.

ztrk, H., **A. zgr**, and **E. Ozkirimli** (2018). DeepDTA: deep drugtarget binding affinity prediction. *Bioinformatics*, **34**(17), i821–i829. ISSN 1367-4803. URL https://doi.org/10.1093/bioinformatics/bty593.