SynerGNN: A Graph Neural Network for Predicting Antibiotic Synergy

By

Colin Small

B.S. Computer Science, University of New Hampshire, 2021

Thesis

Submitted in partial fulfillment of the requirements for the Degree of Master of Science in the Graduate Program of Biotechnology at Brown University

PROVIDENCE, RHODE ISLAND
MAY 2023

AUTHORIZATION TO LEND AND REPRODUCE THE THESIS

As the sole author of this thesis, I authorize Brown University to lend it to other institutions for the purpose of scholarly research.

Date _____ _____

Colin Small, Author

I further authorize Brown University to reproduce this thesis by photocopying or other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research.

Date _____ _____

Colin Small, Author

This thesis by Colin Small is accepted in its present form by the Graduate Program in Biotechnology as satisfying the thesis requirements for the degree of Master of Science

Date _____ Signature: _____

Dr. Lorin Crawford, Advisor

Date _____ Signature: _____

Dr. Jacquellyn Schell, Reader

Date _____ Signature: _____

Dr. Louis Rice, Reader

Approved by the Graduate Council

Date _____ Signature: _____

Dr. Thomas Lewis, Dean of the Graduate School

Acknowledgements

I would like to express my sincerest gratitude to my thesis advisor, Lorin Crawford, for providing me with invaluable guidance, encouragement, and statistical perspectives throughout my research journey. Your patience has been instrumental in helping me navigate the process of conducting rigorous academic research and completing this thesis.

To my housemates and friends at Brown – thank you for your unwavering support and occasional (but quite necessary) distractions which helped me maintain sanity during the stressful periods of my research. Your friendship has been a constant source of comfort and joy, and I will forever be grateful for the memories we created together.

Finally, I would like to express my heartfelt gratitude to my family for their unconditional love and support throughout, before, and beyond my academic journey here at Brown. No amount of words will be able to convey the love and appreciation you all deserve.

And to those above and those unnamed but certainly deserving – thank you all for your contributions, encouragement, and support. I could not have done this without you!

Contents

List of Tables

List of Figures

Abstract

The rise of antimicrobial resistant (AMR) bacteria is one of the world's most pressing public health crises, and our fight against such AMR bacteria is hampered by the time-, labor-, and cost-intensities of discovering novel antibiotics. However, drug repurposing paradigms offer us the opportunity to forgo many of these costs by repositioning existing drugs from their original medicinal purpose into being antibiotic adjuvants, or compounds that when prescribed alongside antibiotics that produce a more potent antibiotic effect than the application of the antibiotic alone. Through the integration of modern deep learning techniques and massive amounts of activity data produced via high-throughput screening of antibiotics and potential adjuvants, we can effectively train models to discovery and predict these novel synergistic antibiotic combinations.

Towards that goal, we propose SynerGNN, a graph neural network model for predicting and discovering novel synergistic antibiotic combinations through screening repurposable molecules. In this work, we evaluate its performance on both antibiotic and non-antibiotic activity datasets and take strides to understand the compounds that it predicted to have the highest antibiotic synergies. We also highlight the challenges of working with limited antibiotic synergy datasets and justify the need for a more complete dataset of antibiotic synergy data the implementation of machine learning techniques that will help us interpret the molecular structures that SynerGNN associates with synergistic activity. Ultimately, models such as SynerGNN will serve as important tools in the continued fight against antimicrobial resistant bacteria in accelerating the screening process of potential antibiotic adjuvants and in

discovering novel mechanisms of synergistic action that can be exploited to completely break through existing bacterial resistance mechanisms.

Chapter 1

Introduction

1.1 Introduction

Since the discovery of salvarsan and penicillin, antibiotics have proved to be one of the most powerful tools in doctors' arsenals in combating infection, disease, and death. Antibiotics alone have undoubtedly contributed to the over 20-year increase in average lifespan in the US since their discovery and to substantial, though uncalculable, increases in lifespan and quality of life elsewhere in the world. But as we develop new antibiotics to combat bacterial infections, so do bacteria evolve mechanisms to resist antibiotic treatment. Patients infected with antimicrobial resistant (AMR) bacteria face much harsher outcomes and treatment regimens, requiring second- or third-line therapeutics. For others, AMR bacterial infections prove so resistant that doctors are reduced to palliative care and treatment with unconventional or experimental treatments [1]. This resistance, and the growing number of global AMR-implicated infections and deaths, is undoubtedly one of the most pressing modern public health crises.

In this thesis we propose SynerGNN, a graph neural network model for predicting and discovering novel synergistic antibiotic combinations through screening repurposable molecules. We start by motivating the need for such a model with a review of antibiotic infection and discovery trends, with a particular focus on synergistic combination therapy, drug repurposing, and virtual screening. We then introduce neural

network concepts to preface a description of the data and computational methods used in SynerGNN. We next evaluate SynerGNN's performance on both antibiotic and non-antibiotic activity datasets and provide a brief on review the compounds combinations that it predicts to have the highest antibiotic synergies. We conclude the thesis by highlighting the challenges of working with limited antibiotic synergy datasets and justifying the need for a more complete dataset of antibiotic synergy data the and for machine learning techniques that will help us interpret the molecular structures that SynerGNN associates with synergistic activity.

1.2 The Antimicrobial Resistance Crisis

1.2.1 Antimicrobial Resistant Infections

In 2019, the United States Centers for Disease Control (CDC) released their Antibiotic Resistance Threats in the United States report outlining trends in the growing domestic antimicrobial resistance crisis, highlighting a 6-year increase in incidence of 200,000 cases which totaled 2.8 million and 35,000 AMR-implicated infections and deaths, respectively, for that year [2]. A similar report published in *The Lancet Infectious Diseases* estimated deaths attributable to and the incidence of antibiotic-resistant bacteria in the the European Union and European Economic Area, finding that the incidence of most AMR strains to be 2.6 times higher in Europe [3] than in the United States. Murray *et al.* [4] provide a further global perspective on the health burden due to AMR infections, combining data from systematic literature reviews, EHR systems, AMR surveillance systems, and other sources covering 23 different pathogens in 204 countries and territories. The authors liken the global health burden of AMR bacterial to HIV and malaria, attributing over 4.95 million deaths globally with AMR infections.

Each reports' findings underscore AMR infections as a clear global health crisis and call on researchers to improve upon currently lacking antibacterial drug development

efforts with novel discovery strategies.

1.2.2 Declining Investment in Antibiotic Development

The striking rise in incidence of AMR infections has not been met with an increase in antibiotic development [5].

Foremost, any novel therapeutic agent must undergo an extensive series of clinical trials to prove both safety and efficacy, typically involving Phase I (assessing safety), Phase II (assessing efficacy), and Phase III (assessing longer-term therapeutic effect) studies. The costs of conducting these clinical trials are immense - the combined mean cost of a single phase I, II, and III study summed to $339.4 million (in 2013 dollars) as estimated by DiMasi *et al.* [6]. Many novel agents will undergo several phase II and III studies, and low clinical success rates demand the running of multiple concurrent investigations for novel therapeutic agents, pushing mean out-of-pocket clinical costs to drug developers for a single successful agent to $965 million. In total, estimated drug development costs exceed $1.3 billion.

These costs are acceptable in drug development efforts for drugs whose sales are expected to far recoup their development costs. Prospective sales for antibiotics, however, are far more limited. Compounding this is the fact that the middle- and low-income countries who are most burdened by the AMR crisis are "in the weakest political and economic position to lead the reform of the antibiotics market" [5]. Considering these development costs and limited financial prospects, the development of a novel antibiotic nets a loss of over $50 million [5] and has lead to the divestment of four major pharmaceutical companies from the antibiotic development space (AstraZeneca in 2016, and Novartis, Sanofi, and Allergan in 2018) and no antibiotics with novel chemical structure or mechanism of action have been brought to market since that time frame [5, 7].

Although recent corporate and corporate-governmental programs such as The AMR Action Fund and CARB-X promise to invest necessary funding in developing novel

antibiotics [8, 9], the financial pressures brought by early phase clinical trials and pre-clinical research remain. Progress in drug development cost reduction must therefore come from novel therapeutic techniques such as combination antibiotic therapy and drug repurposing and innovation in discovery strategies such as virtual screening.

1.3 Combination Antibiotic Therapy

1.3.1 Combination Antibiotics

A combination antibiotic is a joint formulation of several different antibiotics into a single therapeutically deliverable drug and offer multiple advantages over the prescription of single antibiotics alone.

One of the largest advantages to combination antibiotic therapy is synergy. Antibiotic synergy is a phenomenon in which a combination antibiotic produces a greater therapeutic effect than would be achieved by using each constituent antibiotic in the combination alone. In other words, the antibiotics work together to enhance their individual effects, resulting in a more effective treatment against bacterial infections. Other advantages include the ability to break through AMR mechanisms thus restoring an antibiotic's efficacy against otherwise-resistant strains [10–12] and reduced toxicity achieved by using lower-than-normal doses of the constituent antibiotics.

Some combination antibiotics include a non-antibiotic agent (called an adjuvant) that enhances the effects of the one or more antibiotics that are included alongside the adjuvant [13, 14]. Many adjuvants are capable of restoring efficacy to antibiotics that bacteria have developed resistance to. For example, many bacteria that are resistant to the antibiotic amoxicillin have developed special proteins that actively destroy any amoxicillin molecule that enters their cell. The adjuvant clavulanic acid inhibits this protein, thus re-enabling amoxicillin to enter and kill bacterial cells [15].

The main advantage conferred by the inclusion of adjuvants over antibiotic combinations that contain only antibiotics is that adjuvants target novel proteins to which

bacteria have never developed resistance mechanisms.

1.3.2 Drug Repurposing

Drug repurposing is an attractive method of drug discovery allowing researchers to forego many time- and cost-intensive preclinical and early-stage clinical research steps. Unlike de novo drug development, drug repurposing identifies new therapeutic uses for existing drugs that were originally developed for different indications. Drug repurposing allows researchers to leverage existing safety and efficacy data of repurposable drugs to reduce the time- and cost-burdens of de novo drug development.

Brown [16] conducted an extensive literature review and primary interviews with researchers and physicians to highlight a select number of traditionally non-antibiotic drugs with promising characteristics as either antibiotics or adjuvants, including Ciclopirox (an antifungal), Loperamide (an antidiarrheoal), Berberine (a natural product antidiarhheoal and traditional medicine of Europe, Asia, and the Americas), Curcumin (the principal component of Turmeric), Epigallocatechin-3-gallate (a natural product found in green tea), and (+)-Naltrexone and (+)-Naloxone (both opioid antagonists). These molecules were chosen as priority repurposing candidates due to favorable antibiotic or adjuvant activity and published drug safety.

1.3.3 Virtual Screening

Virtual screening is a drug discovery technique for identifying therapeutically active compounds within large libraries of molecules using a variety of computational methods. By screening a vast number of compounds in silico, virtual screening enables the identification of promising leads that can be further evaluated and optimized through experimental assays.

Traditional drug development instead relies on extensive wetlab screening of compounds' activity on chosen targets, often involving extensive plating, incubating, and analysis efforts by scientists in-lab culminating in a handful of promising (or lead)

compounds. Although the advent of high-throughput screening (HTS) allowed for the automation and parallelization of these wetlab tasks, HTS still requires the procurement of library compounds, incubation time, and in some cases manual human review for up to hundreds of thousands of compounds. Virtual screening allows us to offload these steps to computers, where predictions can be run on massive compound libraries at a fraction of the cost and time required even for HTS.

It should be emphasized that virtual screening is not a complete replacement for in-vitro screening (high-throughput or otherwise) but rather is a means to reduce the time- and labor-investments of lead compound identification. Such costs scale with the number of compounds that must be screened in-vitro, and virtual screening allows us to reduce wetlab screening to a limited number of compounds that are most favorably predicted to be therapeutically active.

Examples of drugs developed using virtual screening include Captopril (an antihypertension ACE inhibitor), Dorzolamide (an anti-ocular hypertension drug), Saquinavir (an antiretroviral HIV/AIDS preventative), Zanamivir and Oseltamivir (both anti-influenza medication), Aliskiren, Boceprevir, Nolatrexed, and Rupintrivir, although an extensive list is undoubtedly much larger owing to the widespread adoption of virtual screening by pharmaceutical companies [17].

Most modern virtual screening campaigns utilize some form of machine learning to predict therapeutic activity of candidate molecules.

1.3.4 Machine and Deep Learning for Drug Discovery

Owing to the increasing ease with which we can collect and access massive sets of data, the use of deep learning for predictive and classification tasks has exploded since the mid 2000's. From general speech recognition to financial fraud detection, deep learning has become a staple in computer-assisted and -accelerated tasks. The field of pharmacology is no different, and deep learning has seen extensive use in antibiotic development and drug combination discovery.

Graph-based deep learning approaches specifically have found success in the discovery of novel antibiotics. Stokes *et al.* [18] developed a deep learning pipeline to predict antibacterial activity of single molecules. Their work culminated in the discovery of the novel antibiotic Halicin, repurposed from a molecule previously clinically developed as an antidiabetic medication. In vitro and in vivo assays revealed potent antibacterial activity of Halicin, particularly of note against many AMR strains of bacteria. Halicin, despite having low molecular similarity to the closest equivalent molecule in their training data, was identified by their model as having high likelihood of antibiotic activity. This suggests that the model (and graph-based deep learning approaches in general) are able to identify molecular substructures indicative of inhibitory growth activity outside of the known traditional antibiotic chemical space. Indeed, further analysis of Halicin revealed a novel mechanism of antibiotic action - interfering with cellular ability to regulate pH balance across the cell membrane - that is integral to its ability to fight AMR strains.

For a more extensive list of antibiotic-discovery deep learning applications, David *et al.* [19] provide a review of existing AI applications to this task.

Similar computational architectures have been successfully used in predicting synergistic therapeutic combinations in diseases ranging from COVID-19 to cancer. Jin *et al.* [20] extended the deep learning framework used in Stokes *et al.* [18] to investigate the role of multiple molecules in synergistic COVID-19 treatments. In addition to treating each molecule as its own graph structure, the authors developed a secondary drug-target interaction model to predict the affected pathways of each drug in a combination. The predicted targets were fed through a target-disease association layer to further predict the affect of the target on COVID-19 disease treatment. Individual single-agent activity and predicted combination activity were used to calculate the combination's Bliss score [21], indicating potential synergistic activity.

Similarly, Preuer *et al.* [22] developed a deep learning model to predict novel synergistic cancer therapeutics. DeepSynergy is a dense, feed forward neural network

incorporating both the chemical descriptors of the drug combination and genomic information of the targeted cell line. Owing to limitations in their training data, the authors found their model struggled to generalize across novel drugs and cell lines but performed well on predicting novel drug-cell line combinations where both the drugs and cell lines were included in the training data.

1.4 Conclusion

The rise of antimicrobial resistant (AMR) bacteria is one of the world's most pressing public health crises, and our fight against such AMR bacteria is hampered by the time-, labor-, and cost-intensities of discovering novel antibiotics. However, drug repurposing paradigms offer us the opportunity to forgo many of these costs by repositioning existing drugs from their original medicinal purpose into being antibiotic adjuvants, or compounds that when prescribed alongside antibiotics that produce a more potent antibiotic effect than the application of the antibiotic alone. Through the integration of modern deep learning techniques and massive amounts of activity data produced via high-throughput screening of antibiotics and potential adjuvants, we can effectively train models to virtually screen novel synergistic antibiotic combinations. We thus propose SynerGNN, a graph neural network model for predicting antibiotic synergy and discovering novel synergistic antibiotic combinations through screening repurposable molecules.

Chapter 2

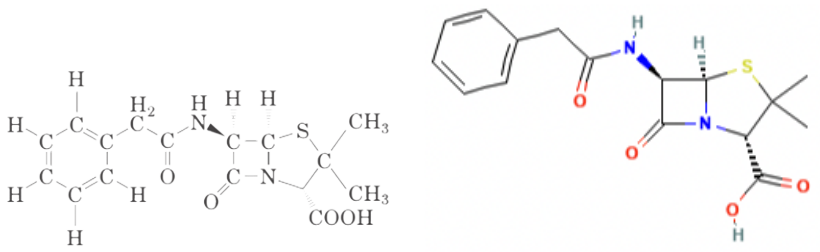
Neural Networks for Molecular Learning Tasks

2.1 Background and Motivation

As touched on in chapter 1, machine learning models vary widely in design, implementation, and task. In the context of this thesis, we wish to design a machine learning model that can predict, given two molecules, any possible antibiotic synergy that those molecules may have. At it's core, however, it easier to think of this task as a general problem of molecular property prediction. That is, we want to feed our model a datapoint consisting of one or more molecules and have it output some predicted numerical property of that datapoint, whether that property be antibiotic activity for a single molecule, antibiotic synergy in the case of multiple antibiotics, or some other property like lipophilicity (i.e. how easy it is for a molecule to pass through cellular membranes). To accomplish this task, we require: 1) a method of representing molecules, and 2) a model for generating a prediction given that representation.

2.2 Representing Molecules

Molecules consist of atoms and the bonds that connect them. On paper, molecules are often represented by their Lewis structure or skeletal formula, which shows in two dimensions the atoms, denoted by their chemical symbol, and bonds, denoted by lines representing either single, double, or triple bonds.



(a) (b)

Figure 2.1: Examples of (a) Lewis structure [23] and (b) skeletal formula of Penicillin.

Lewis structures and skeletal formulas are graphical and intuitive — a human chemist can easily understand molecules represented these ways. But while some progress has been made in using these graphical depictions of molecules for computerized tasks of predicting biochemical activity and other properties [24], computers generally require molecules to be represented in numbers and bits rather than images. One such way to do so is by using graphs.

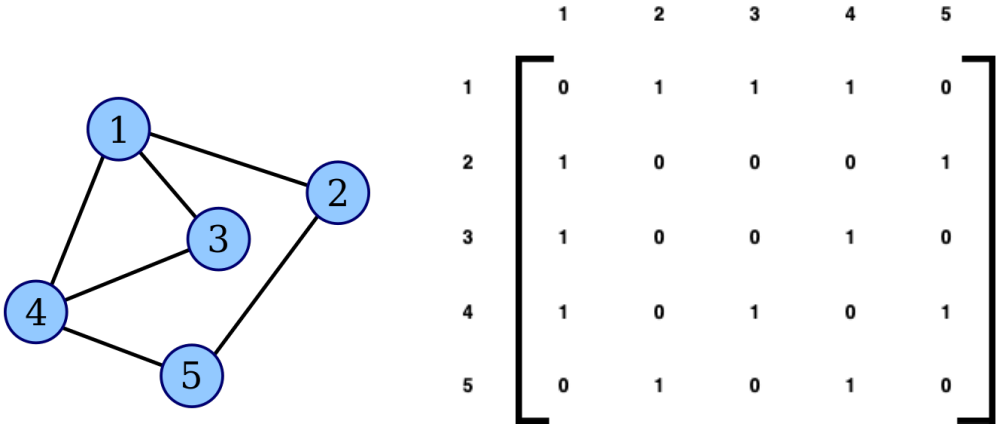
2.2.1 Graph Data Structures

Graph data structures are a way to represent a set of entities or objects and the connections between them. In a graph, objects are represented as vertices (also known as nodes), and connections between them are represented as edges. Examples of graphs include social networks, where a set of individuals may be connected by their friendships, or molecules, where a set of atoms are connected by bonds.

In order to be understood by a computer, a graph must be transformed into an adjacency matrix. An adjacency matrix is a tabular representation of a graph. The rows and columns of the matrix represent the graph's nodes, and the values in the matrix represent the edges. If there is an edge between two nodes, the corresponding entry in the matrix is set to a non-zero value, while zero values indicate the absence of an edge. Figure 2.2 shows an example graph data structure shown pictorially and as an adjacency matrix.

The information contained within the pictorial depiction of a graph (Figure 2.10a)

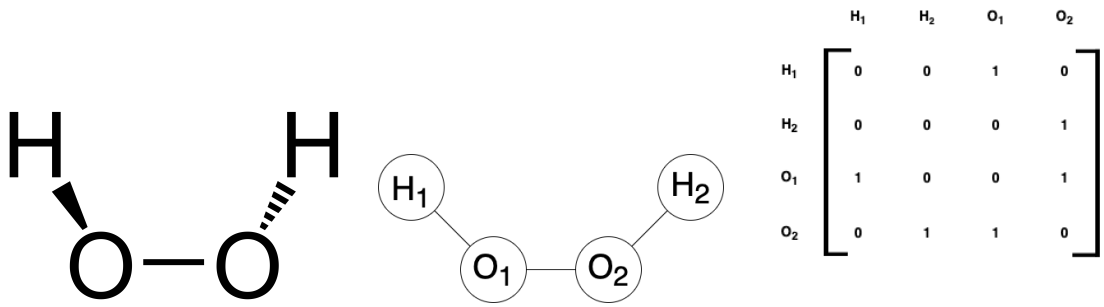
and the adjacency matrix of a graph (Figure 2.10b) is exactly the same. All we have changed is how that information is represented, from a way that is intuitive to human eyes to a way that is intuitive for computer processors to understand.



(a) (b)

Figure 2.2: Example (a) graph data structure and (b) its accompanying adjacency matrix. An entry set to 1 in the adjacency matrix indicates that nodes of that cell's column and row are connected by an edge.

Using graph data structures, we can very easily represent molecules as an interconnected collection of atoms. When creating a molecule graph, we treat each atom as a node and each bond as an edge. Figure 2.3 shows the skeletal formula, pictorial depiction, and adjacency matrix for a molecule of hydrogen peroxide (H_2O_2).



(a) (b) (c)

Figure 2.3: (a) Skeletal formula, (b) pictorial graph, and (c) adjacency matrix (c) of H2O2/Hydrogen Peroxide.

Atom Features

In addition to representing the structure of molecules, we need to be able to describe their contents and make up. We can do this using a set of descriptors, or simple questions that we can ask of each atom in a molecule which may be helpful in accomplishing the prediction task. For example, we can ask what the atomic weight is of each atom in the molecule (hydrogen's is 1.008u and oxygen's is 15.999u). We can also ask true-false questions such as "is the atom a metal atom?" (neither hydrogen nor oxygen are metals). Answers to these kinds of questions are encoded with ones and zeros representing true and false, respectively. For categorical questions such as "what element is the atom?", we ask a series of questions "is the atom a hydrogen?", "is the atom helium", etc., all the up up to "is the atom oganesson?" (the most recently discovered element).

For example, say our atomic descriptors are "what is the atom's atomic weight?", "is the atom part of a ring structure in the molecule?", "is the atom a metal?", and "what is the atomic number of the molecule?". For the atoms in the hydrogen peroxide example above, the answers to these descriptors are the following:

	Atomic Weight	Is part of ring structure?	Valence	Is metal atom?	Atomic Number 1	Atomic Number 2	...	Atomic Number 8	...	Atomic Number 118
H_1	1.008	0	1	0	1	0	...	0	...	0
H_2	1.008	0	1	0	1	0	...	0	...	0
O_1	15.999	0	2	0	0	0	...	1	...	0
O_2	15.999	0	2	0	0	0	...	1	...	0

Figure 2.4: Example atom feature matrix of H2O2/Hydrogen Peroxide. A single row from this atom feature matrix is an atom's feature vector.

The collective set of descriptor answers for a given atom in a molecule is called the atom's feature vector, and the set of feature vectors for all atoms in a molecule can be stacked to form the molecule's feature matrix. When fed to the model, this atom

feature matrix sufficiently describes the contents and make-up of the molecule.

2.3 Neural Networks

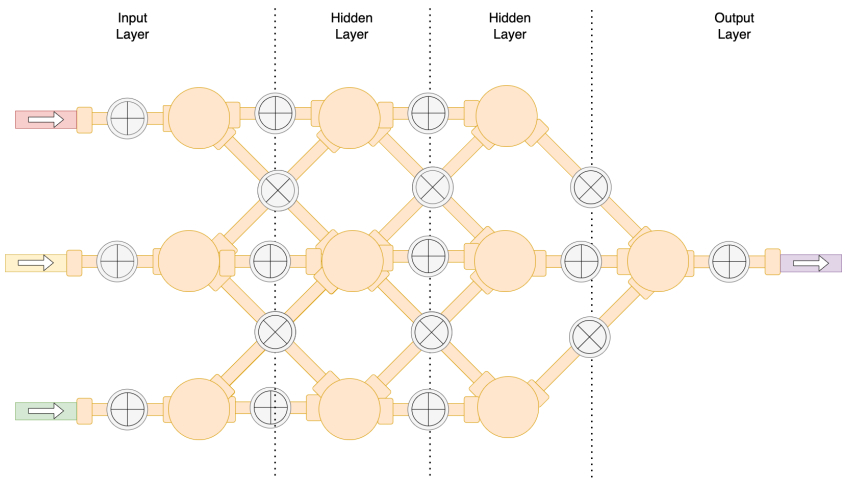
The Pipe Analogy of Neural Networks

A neural network can be thought of as a set of interconnected pipes that process information. Just as water flows through a system of pipes, so do data through a neural network.

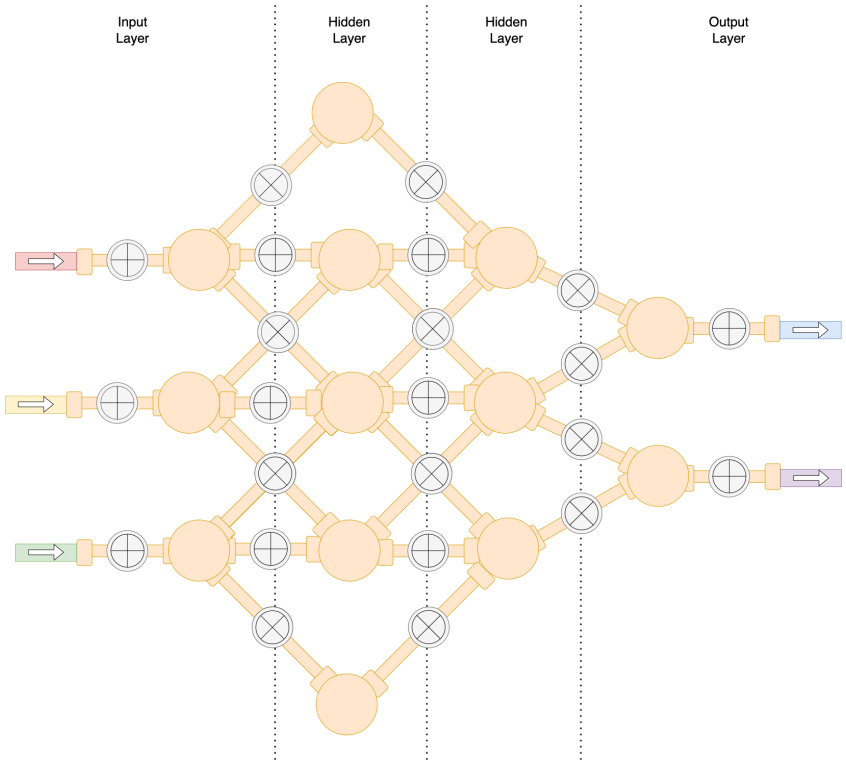
Each pipe has a valve which controls the strength of the water flowing through it, and at the junction of pipes exists a box which carefully mixes the water fed to it by each pipe and outputs a single new stream of water. At the beginning of the pipe system there exists multiple input pipes, each taking in a different stream of water. And at the output of the pipe system there exists a single output pipe, flowing from it a single stream of water that has been carefully mixed and created from the input pipes.

The number of input streams, number of output streams, and placement of the pipes and mixing boxes can be configured in a way that can best accomplish some task. In the case of our pipe analogy, we may design a series of interconnected pipes to mix different input streams of water and output a single, new stream of water in the color of our desire. The model will be built and trained to consistently output certain colors given an input. For example, we may want the pipe system to always output the color purple given input colors of red, yellow, and green, or the color blue given input colors of red, orange, and yellow. The choice of input streams, output streams, pipes, and mixing boxes is called the model architecture, and model architectures can vary according to the required task. Figure 2.5b shows an alternate model architecture with added pipes and mixing boxes and a second output stream for a task that requires us to output two colored streams of water.

Furthermore, within both example models, the pipes and mixing boxes are separated into different layers: an input layer consisting of the first pipes and mixing



(a)



(b)

Figure 2.5: (a) Pipe analogy of a neural network. Three input streams of colored water are fed through a series of interconnected pipes. Each pipe has a valve controlling the strength of the water passing through the pipe, and are joined at junctions that carefully combine the water streams. A single pipe outputs a new stream of water in a color of our desire. (b) An alternative architecture for the pipe analogy, with added pipes and mixing boxes and a second output stream.

boxes that the input streams are connected to, two layers that mix the outputs of the input layer, and an output layer consisting of the single mixing box and pipe leading to the output of the model. The layers in between the input and output of the model are referred to as "hidden layers", since the state of their data (in this case the color of their water streams) is not outwardly visible (conversely, the input and output layers are not hidden layers because we easily know the values of the model at those layers). See Figure 2.5 for a visual aid demarcating the different layers.

For our pipe model's task of creating a certain color given three input color streams, training can be thought of as adjusting the valves in the pipes to achieve a desired flow rate. These valves control the amount of water flowing through a pipe, and, during training, the positions of the valves are adjusted to minimize the difference between the color we want the model to output and the color the model is currently outputting. The ultimate goal is to adjust the valves such that the pipes achieve the correct flow rate into each mixing box to produce the desired color. By repeating this process over and over again for different input streams and desired output colors, we can figure out a single configuration of valve positions that is able to optimally reproduce each desired color. For example, we will find a single valve configuration such that the input colors of red, yellow, and green output purple while the colors red, orange, and yellow output blue. With enough training examples (a set of input colors and the desired output color), we will adjust the valves to positions such that the model will be able to accurately predict output colors given combinations of input colors it has not yet seen in the training process.

Neural Networks as Mathematical Functions

Rather than operating on streams of water, neural networks work with numbers. In neural network parlance, the mixing boxes of our pipe analogy are called nodes, each of which applies a mathematical function to its inputs and outputs a single new value. The pipes with valves connecting the mixing boxes are connections between

each node which scale the output of the node, just as the valves control the flow rate of the water.

Figure 2.6 illustrates an example of a simple neural network architecture for a model that predicts housing sale prices. It takes as input three numbers — the number of bedrooms in the house, the ZIP code the house is located in, and the square footage of the house — and outputs a single number — the final sale price of the home. The model consists of an input layer with three nodes, two hidden layers, each with three nodes, and an output layer with a single node.

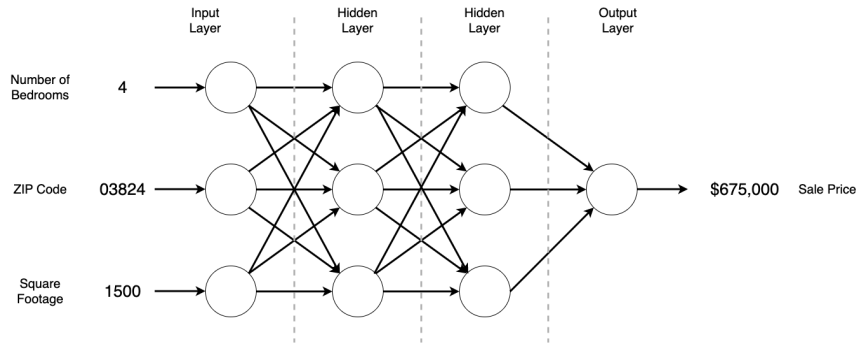


Figure 2.6: An example neural network architecture built to predict the price of houses.

As mentioned above, each node is a mathematical function that transforms and scales its input. The most common implementation of a node multiplies each of its inputs by a scaling factor, (called a weight) adds a constant value to the sum of the scaled inputs (called a bias) and feeds the resulting value through a non-linear function (called an activation function). The specific choice of activation function varies from model to model, and usually model designers test several choices of activation function to find the one that works best for their model's task. Common activation functions include the sigmoid (or logistic) function, the tanh function, and the rectified linear (ReLu) function. The activation function is example of a hyperparameter, or a parameter that model designers choose empirically rather than being tuned automatically in the NN training process. The formula for such a node is as follows:

$$y = f(\sum_{i}^{\#\text{ inputs}} (x_i w_i) + b) \tag{2.1}$$

Where y is the output of the node, f is an activation function, x_i is the i-th input to the node, w_i is the weight associated with the i-th input to the node, and b is the bias of the node. For nodes in the input layer of our house sale price example, the nodes have only a single input — the number of bedrooms, ZIP code, or square footage of the house. Thus, the formula for the number-of-bedrooms input node would be the following:

$$y_{\text{beds}} = f((\text{number of beds} * w) + b) \tag{2.2}$$

Each of the nodes in the first hidden layer are connected to the nodes in the input layer, thus they take as input the output of the nodes in the input layer. Their formula is the following:

$$y = f((\text{number of beds} * w_{\text{beds}} + \text{ZIP} * w_{\text{ZIP}} + \text{footage} * w_{\text{footage}}) + b) \tag{2.3}$$

Every node in the model learns its own weights and biases for each of its inputs. During model training, the weights and biases of each node are adjusted, just as we talked about adjusting the valves in the pipe analogy. The individual weights and biases for each node in the network are called the parameters of the model.

Latent Representations

An important concept in neural networks is the idea of latent representations. A latent representation is simply the outputs of a hidden layer in a neural network given some input, before those outputs are fed to either another hidden layer or the output layer. Let's use our house price model as an example and assume the model has been trained with parameters such that it can perfectly predict the sale price of a home that has not yet sold. If we input the values $\{4, 03826, 1500\}$, representing the number of

bedrooms, the ZIP code, and square footage of the house, respectively, into the model, then each of the nodes in the first hidden layer will output some number following equation 2.3. For example, the outputs of the nodes may be $\{0.65, 0.23, 0.84\}$. This set of numbers is the latent representation of the model after the first hidden layer.

Another helpful analogy to understand latent representations is to imagine the process of baking a cake. The inputs to a cake-baking model might be flour, sugar, and eggs, and the output of the model would be a finished cake. The latent representations of the model are thus the different stages of combining and baking the cake — the latent representation of the first hidden layer in the model might be the ingredients combined in a mixing bowl, and the latent representation of the last hidden layer might be the cake batter right before it is finished baking. At each of these stages, the combination of ingredients is not yet a cake but is in the process of becoming one. Even the slightest difference in the amount of ingredients used will alter the consistency of the cake batter, and the slightest difference in cake batter consistency will alter how well the cake sets in the oven, and the slightest difference in how the cake sets in the oven will alter the final cake that is made. A certain batter consistency can only be achieved with certain ingredient amounts and will always lead to the same final cake, just like how the latent representation of $\{0.85, 0.11, 0.97\}$ in our housing model will always lead to a predicted sale price of $\$675,000$.

2.3.1 Graph Neural Networks

For data that can be represented with graph structures, we can use graph neural networks (GNNs) to greatly improve predictive performance on tasks by utilizing the structural information contained within the graphs.

Say we wish to expand our home price model to now model the attractiveness of a neighborhood given the characteristics and layout of the houses in that neighborhood. We could simply build a larger basic neural network with inputs representing *(bedrooms in House A, square footage of House A, bedrooms in House B, square footage*

of House B, etc.), but such a network would have two critical drawbacks. First, the network would have no concept of locality for the neighborhood — it wouldn't know which houses are located next to which. Second, because neural network architecture (including the number of inputs to the model) is held static after we commence training, we would need to create and train a separate model for every conceivable number of homes that could exist in a neighborhood (e.g. a 5-home neighborhood model, a 6-home neighborhood model, etc.).

Instead, we can build a graph neural network that addresses both of these limitations. We first model the neighborhood as a graph, treating each house as a node and connecting the house-nodes with edges representing the presence of a shared property line between the two houses (i.e. the properties of the two houses are adjacent to each other). For each house, we also create a feature vector containing three descriptors: the number of bedrooms in the house, the number of bathrooms in the house, and the square footage of the house. This graph structure, and the feature vectors of the nodes in the graph, forms the input to our GNN. An example neighborhood graph is shown in Figure 2.7.

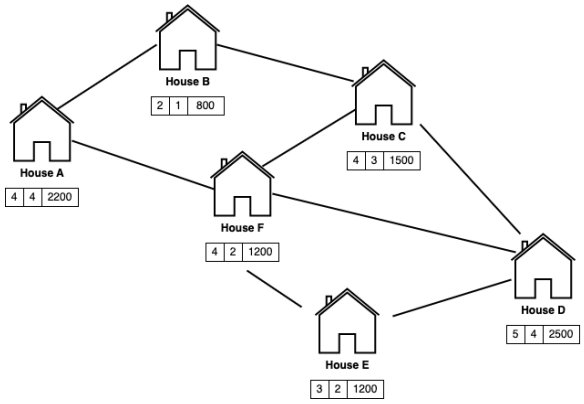


Figure 2.7: An example neighborhood represented as a graph. In this example, an edge between two house nodes represents that the properties of those houses share a border (e.g. the property of House F abuts the properties of Houses A, C, D, and E.). Each house has an associated feature vector listed underneath its label containing the number of bedrooms in the house, the number of bathrooms, and the square footage. For example, House A has 4 bedrooms, 4 bathrooms, and has 2200 square feet of livable space.

After our graph and feature vectors have been created, the internal workings of the GNN can be broken down into five steps (visualized in Figures 2.8 and 2.9):

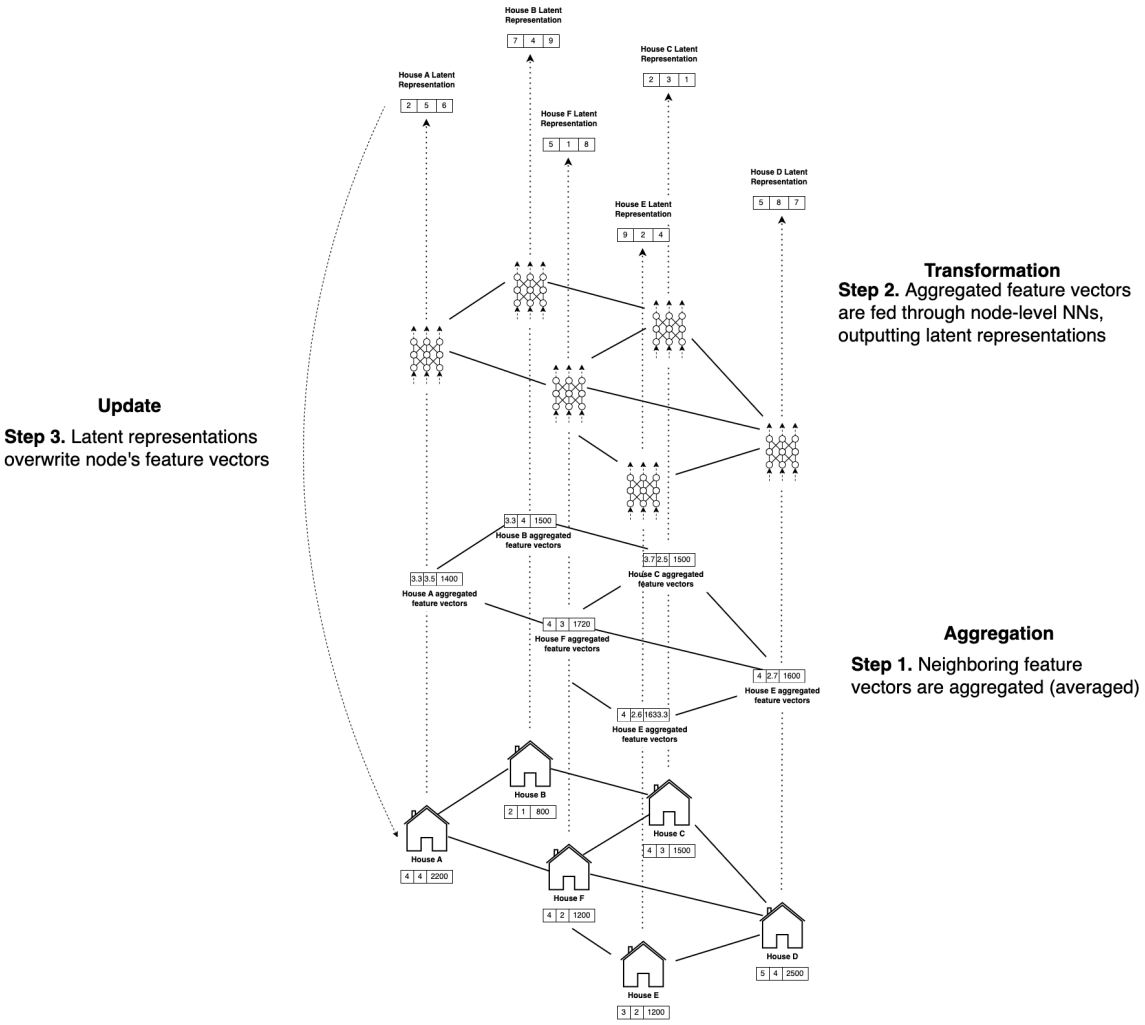


Figure 2.8: Graphical depiction of the aggregation-transformation-update process of a graph neural network.

1. **Aggregation:** The first step of the GNN is aggregation. During this step, each node aggregates the feature vectors of its neighbors. This aggregation can be performed in various ways, such as taking the mean or sum of the neighboring node features. The goal of the aggregation step is to combine the information from the neighbors of a node in a way that is informative for that node's own representation. Each node only aggregates the feature vectors of its direct

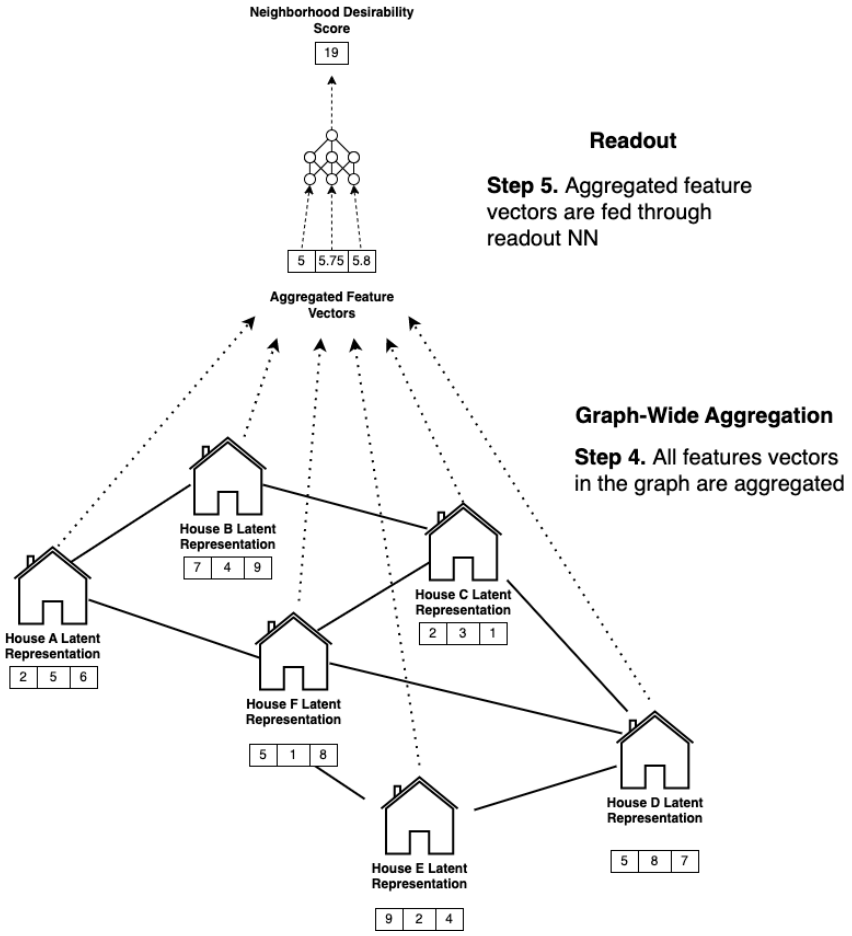


Figure 2.9: Visual depiction of the graph-wide aggregation and readout process of a graph neural network.

neighbors. For example, House A will only aggregate together its feature vector and the feature vectors of Houses B and F.

2. **Transformation:** After the first step, each node is left with a feature vector that represents the aggregated features of itself and its neighbors. This feature vector is then fed through a simple neural network, called a node-level neural network because it operates on the feature vectors of each node. The architecture of this internal network will vary from GNN to GNN, but is always set up to take as input as many values that are stored in the feature vectors of the nodes and to output a vector containing a latent representation of the neural network (see the definition of latent representations in the prior section). In the

graphic example of this process in Figure 2.8, each house node's feature vector is fed to a different node-level neural network, but in practice there is only ever a single node-level neural network that is shared by all of the nodes.

3. **Update:** After latent representations have been generated for each node in the network, the update step overwrites the feature vectors of each node with their corresponding latent representations. The update step is a crucial part of the GNN because it allows the model to learn from the information that has been aggregated and transformed, and update the state of each node based on that information. This step represents the model furthering its understanding of each node. It is vital to note that because each node aggregated its neighbors' feature vectors before transformation, that latent representation that is assigned to each node during the update step contains some amount of information about its neighbors.

   Steps one through three are then repeated a user-defined number of times, and it this iterative process of aggregation, transformation, and update that enables the GNN to meaningfully understand the graph structure and nodes for downstream tasks. For example, after the first iteration of aggregation-transformation-update, the feature vector of House C contains information from Houses B, F, and D. Because House C is not a direct neighbors of Houses A and E, it has not yet had a chance to learn about the entire graph. However, during the first iteration of this process, House F updated its own feature vector with information from Houses A and E (alongside Houses C, D, and E), meaning its feature vector contains some information about those houses. (Likewise, after the first iteration House A's feature vector now contains information about Houses B and F, E's contains information about F and D, and so on.) Thus, during the second iteration of the aggregation-transformation-update process, when House C aggregates the feature vector of House F, it is really pulling

in some information about Houses A and E, since although C is not a direct neighbor of those houses and cannot access their feature vectors, House F has.

The number of times we repeat this process is called the graph neural network's depth. For example, a GNN with depth = 3 would repeat the aggregation-transformation-update process three times. Like the activation functions of the model, the exact depth is a hyperparameter and is usually set after testing several different values and determining which leads to the best performing model.

4. **Graph-Wide Aggregation:** After the aggregation-transformation-update process has been repeated the desired number of times, the model aggregates the feature vectors of all nodes in the network into a single graph-wide aggregated feature vector. Like the aggregation step before, the exact way these vectors are aggregated can vary according to model designer's preference (mean, sum, max, etc.). This vector represents the latent representation and the model's understanding of the entire graph.

5. **Readout:** Finally, the graph-wide aggregated feature vector is fed into a final basic neural network, called the readout neural network. This neural network uses the amassed understanding of the graph and its contents (as represented by the aggregated latent feature vector) to predict some value. In our example, it uses what the node-level neural networks have learned about the individual houses in the neighborhood and the shape of the neighborhood to output some value representing its desirability.

The training process for GNNs is largely the same as described for basic neural networks. By showing the graph neural network many different training examples, it is able to slowly tune the weights and biases in both the node-level neural network and readout neural network to accurately predict unseen training data.

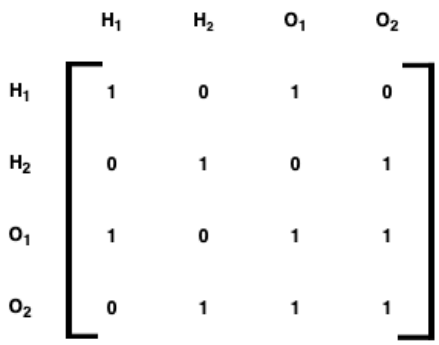
Mathematical Description of Graph Neural Networks

In order to describe graph neural networks mathematically, we will switch our example away from a housing price model to building a model to predict properties of molecules. Mathematically, the model takes advantage of the adjacency matrices and feature matrices we introduced in Section 2.2.1. The adjacency matrix is denoted as a matrix $A \in R^{N_n \times N_n}$, where N_n is the number of nodes in the matrix, and the atom feature matrix is a matrix $X \in R^{N_n \times N_f}$ where N_n is again the number of nodes in the matrix and N_f is the number of features we have described/created for each node. In our H_2O_2 example, there are a total of four nodes in the graph, and we have created a total of 122 features per node (atomic weight, ring structure, valence, "is metal?", and "is atomic number X" for each of the 118 known elements), thus $A \in R^{4\times4}$ and $X \in R^{4\times122}$. We also self-connections to the nodes in the adjacency matrix to allow the node to aggregate its own feature vector in addition to its neighbors by setting the diagonal of the adjacency matrix to one.

Before the aggregation-transformation-update process, the input atom feature matrix is projected to a desired latent representation size h (called the hidden size) by passing it through an input node-level neural network: $X^0 = f_{\text{input}}(X)$, where f_{input} is a simple neural network ($f_{\text{input}} : R^{N_n \times N_f} \to R^{N_n \times h}$) and X is the unprojected, raw atom feature matrix described in the paragraph above. The latent representation size h is a hyperparameter and is chosen empirically by the model designer — several choices of h are tested to determine which produces the best-performing model.

1. **Aggregation:** As mentioned in the description of aggregation above, this step can be implemented in multiple ways, typically taking by the average of neighboring feature vectors or the sum of neighboring feature vectors.

   Calculating the sum of the adjacent feature vectors for each node can be accomplished efficiently in a single matrix multiplication step: $X_{\text{agg}} = AX$, where $X_{\text{agg}} \in R^{N_n \times h}$ is the matrix representing the aggregated feature vectors for each



(a)

	Atomic Weight	Is part of ring structure?	Valence	Is metal atom?	Atomic Number 1	Atomic Number 2	...	Atomic Number 8	...	Atomic Number 118
H_1	1.008	0	1	0	1	0	...	0	...	0
H_2	1.008	0	1	0	1	0	...	0	...	0
O_1	15.999	0	2	0	0	0	...	1	...	0
O_2	15.999	0	2	0	0	0	...	1	...	0

(b)

Figure 2.10: (a) Adjacency matrix A for H_2O_2 (b) Atom feature matrix X for H_2O_2.

node.

Calculating the average is a simple extension of taking the sum: $X_{\text{agg}} = ((AX)^T/s)^T$, where $(AX)^T$ denotes taking the transpose of the matrix multiplication of A and X, / denotes element-wise division between the matrices (i.e., divide each row of the matrix $(AX)^T$ by the corresponding row of the vector S), and $s \in R^{N_n}$ is a vector representing the row-wise sum of the adjacency matrix A. In other words, we first calculate the sum of the adjacent feature vectors for each node then divide each sum by the number of neighbors that that sum's node is connected to.

After aggregation, X_{agg} contains, in each row, the aggregation of the feature vectors that that row's node is connected to.

2. **Transformation:** The transformation step involves passing each aggregated feature vector through the node-level neural network. Let f_{node} be a simple neural network that functions as defined in Section 2.3 that takes as input a node's current latent representation vector and outputs an updated latent representation vector for that node: $f_{\text{node}} : R^h \rightarrow R^h$, where h is the hidden size of the latent representation. The transformation step is then simply $X_{\text{trn}} = f_{\text{node}}(X_{\text{agg}})$, where f_{node} is applied to each row of X_{agg}. The resulting output is collected in the corresponding row of $X_{\text{trn}} \in R^{N_n \times h}$, and X_{trn} is the transformed feature matrix.

3. **Update:** In the update step, we simply overwrite the original X with X_{trn}: $X \leftarrow X_{\text{trn}}$. This sets us up to repeat steps one through three any number of times.

   If we assume our model uses sum aggregation, the above steps can be summarized as: $X^{i+1} = f_{\text{node}}(AX^i)$ repeated from $[0...d]$, where X^i is the values of the feature matrix at depth i, i is the current depth of the model, and d is the specified maximum depth of the model. X^0 is the projected input matrix. Likewise, if our model uses mean aggregation, the steps are summarized: $X^{i+1} = f_{\text{node}}(((AX^i)^T/s)^T)$.

   It is very common to see implementations of graph neural networks that train separate node-level neural networks for each layer of depth in the GNN. This allows each layer of depth in the GNN to independently learn the importance of different features. For these types of models, we will have d different node-level neural networks: $\{f^i_{\text{node}} | i \in [0...d]\}$. Thus, the sum aggregation model becomes $X^{i+1} = f^i_{\text{node}}(AX^i)$ and the mean aggregation model becomes $X^{i+1} = f^i_{\text{node}}(((AX^i)^T/s)^T)$.

4. **Graph-Wide Aggregation:** After the aggregation-transformation-update steps are repeated d times, we are left with a feature matrix $X^d \in R^{N_n \times h}$. This feature

matrix must then be reduced to a single aggregated feature vector $v_{\text{agg}} \in R^h$. In other words, we need to aggregate each feature descriptor in some way for all nodes in the matrix.

To calculate the sum of each feature descriptor for all nodes, we calculate: $v_{\text{agg}} = \sum_{i=1}^{N_n} X_{\cdot i}^d$ where $X_{\cdot i}^d$ denotes the i-th column of X^d.

To take the mean of each feature descriptor for all nodes, we can calculate $v_{\text{agg}} = (1/N_n) * \sum_{i=1}^{N_n} X_{\cdot i}^d$ where $X_{\cdot i}^d$ denotes the i-th column of X^d, and the sum is taken over all columns of X. The $(1/N_n)$ term scales the sum to compute the mean, where N_n is the number of columns/features in X^d.

Like the node-level aggregation step, there are other methods to aggregate a matrix which are not listed here.

5. **Readout:** After aggregating the features of each node, we are left with a single aggregated feature vector $v_{\text{agg}} \in R^h$, which represents the summarized knowledge built up by the model of each feature across the entire graph. We then let f_{readout} be a simple neural network that functions as defined in the section *Neural Networks as Mathematical Functions* above that takes as input the aggregated feature vector and outputs a single value representing the property of the molecule we wish to predict: $f_{\text{readout}} : R^h \to R$. To calculate the property we wish to predict, we simply feed the aggregated feature vector into the readout neural network: $\hat{y} = f_{\text{readout}}(v_{\text{agg}})$, where $\hat{y}$ is the predicted property.

Graph Neural Networks for Multiple Molecules

Expanding the above description of graph neural networks to take multiple graphs as input is straightforward. In the context of this thesis, we wish to create a graph neural network that can take as input two molecules and output their predicted antibiotic synergy.

To do so, we make several changes to the model described above:

1. We replace the readout neural network f_{readout} described above with a neural network that outputs a latent representation vector $\overrightarrow{z} \in R^h$. $f_{\text{latent}} : R^{N_f} \rightarrow R^h$. Thus, instead of the process above outputting a single numerical value (the value we wish to predict), it outputs a latent representation of the molecule that has been passed through it.

2. We take the two molecules for which we wish to predict antibiotic synergy for and run them through the GNN process separately, resulting in two latent representations, $\overrightarrow{z_1}$ and $\overrightarrow{z_2}$ for the first and second molecule in the pair, respectively.

3. We concatenate the two latent representations $\overrightarrow{z_1}$ and $\overrightarrow{z_2}$ into a joint latent representation $\overrightarrow{z_{\text{joint}}} \in 2 * R^h$. (Put mathematically, $\overrightarrow{z_{\text{joint}}} = \overrightarrow{z_1}||\overrightarrow{z_2}$)

4. We create a final readout neural network that takes as input the concatenated latent representation vector $\overrightarrow{z_{\text{joint}}}$ and outputs a single numerical value representing the joint property of the molecules we wish to predict, $f_{\text{readout}} : 2 * R^h \rightarrow R$.

5. We pass the joint latent representation $\overrightarrow{z_{\text{joint}}}$ through the final readout neural network: $\hat{y} = f_{\text{readout}}(\overrightarrow{z_{\text{joint}}})$, where $\hat{y}$ is the predicted property.

These changes, and the overall dataflow for a graph neural network that takes as input multiple molecules is summarized in Figure 2.11.

This chapter describes the common implementation of graph neural networks and does not represent any novel contribution to the field of machine learning. However, in Section 3.2, we will describe the implementation of attention and co-attention functions which are contemporary, advanced techniques used to improve GNN predictive performance.

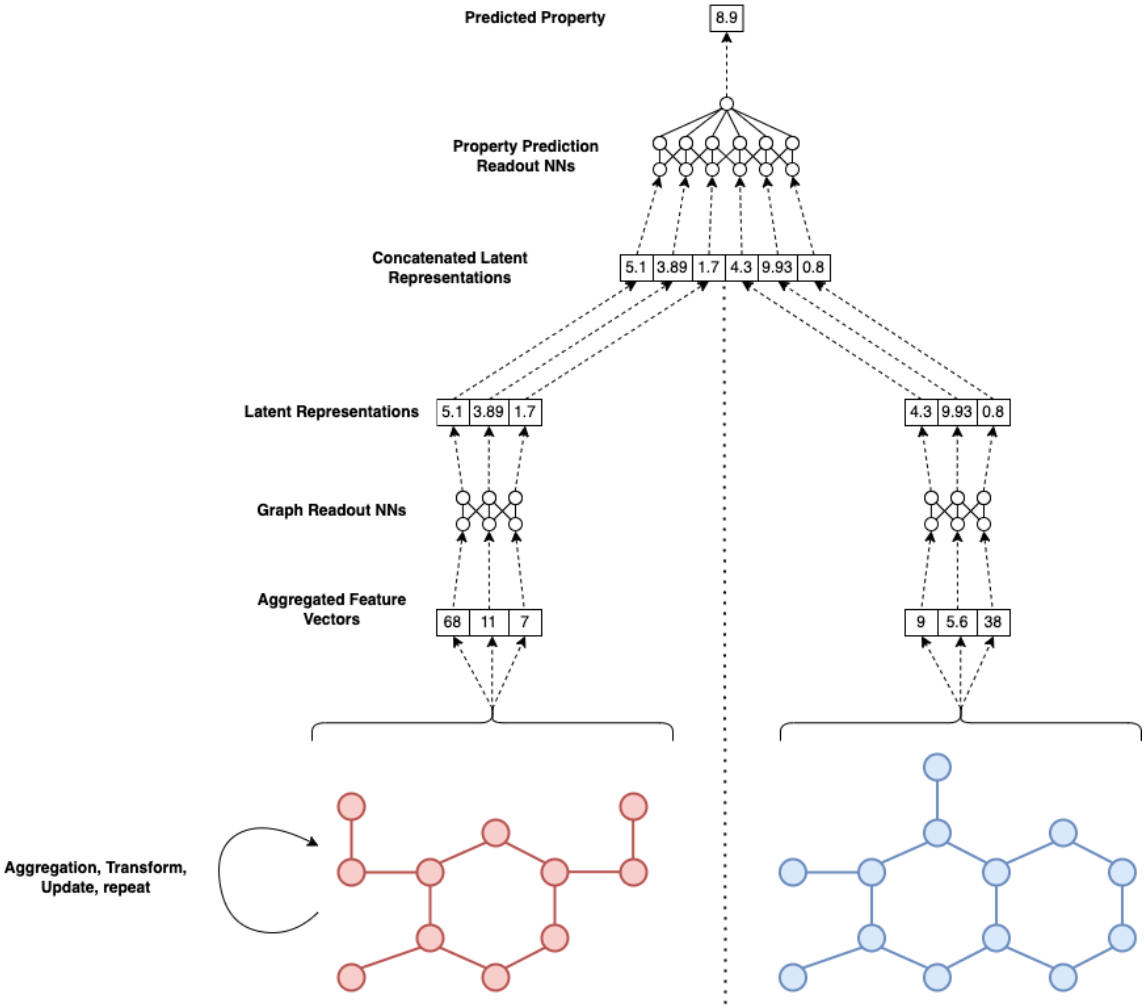


Figure 2.11: Visual depiction of the readout process for a graph neural network that takes two graphs as input.

Chapter 3

Data, Methods, and Computational Details

3.1 Data

For the purpose of antibiotic adjuvant discovery and in order to benchmark against other state-of-the-art molecular property prediction models, we evaluated SynerGNN on several datasets.

3.1.1 Multi Molecule Antibiotic Combination Data

Antibiotic combination data was sourced from work by Kulesa *et al.* [25], who developed a microfluidic high-throughput combinatorial drug screening platform. In their work, the authors screened 4,000 investigational and approved drugs (adjuvants) against a panel of 10 commonly prescribed antibiotics. Each combination (composed of a single adjuvant and a single commonly prescribed antibiotic) was evaluated in a fluorescence-based growth-inhibition phenotypic screening assay to determine growth-inhibitory ability against *Escherichia coli*. Further, growth-inhibitory ability of each combination was evaluated at three dose levels with and without the constituent adjuvant in order to establish dose-response relationship and potential synergy. Synergy was quantified with Bliss scores [26], calculated as the deviation from Bliss independence: Bliss $= f_{AB} - (f_A + f_B - f_A f_B)$, where f_{AB} was the growth inhibition rate of the antibiotic and adjuvant in combination, and f_A and f_B were the growth inhibition

rates of the antibiotic alone and the adjuvant alone, respectively. In all, the authors supplied Bliss scores for a total of 27,600 adjuvant-antibiotic combinations.

For the purposes of antibiotic discovery in this thesis, we primarily trained SynerGNN on the task of predicting the known Bliss scores for these 27,600 combinations. To fairly evaluate SynerGNN's performance on unseen drug combinations, 20% of the 27,600 combinations in the Kulesa *et al.* [25] dataset were randomly held out of the training process at runtime. We evaluated SynerGNN's performance on the remaining 5,520 combinations after each epoch of training to establish the model's performance.

3.1.2 Single Molecule Lipophilicty Data

In addition to the antibiotic combination datasets above, we evaluated SynerGNN on a dataset containing known experimentally-derived lipophilicty values for single molecules [27]. A molecule's lipophilcity impacts many properties of that molecule, including its solubility and membrane permeability. It is well understood that the structural makeup of molecules leads way to their lipophilcity, making this dataset an excellent benchmark to evaluate a model's ability to understand structural-property relationships of single molecules. In total, the dataset contains octonal/water distribution coefficient values (pH 7.4 log D) for $4,200$ different compounds.

3.1.3 Drug Repurposing Data

A core goal of this thesis is to investigate and discover novel synergistic antibiotic combinations using repurposable drugs. To do so, we utilized the Broad Institute's Drug Repurposing Hub [28], a publicly-accessible curated collection of FDA-approved and investigational compounds. These unique compounds provide us with a set of adjuvants that we can computationally screen alongside the 10 commonly prescribed antibiotics used in the Kulesa *et al.* [25] dataset.

To assemble the drug repurposing dataset, we created pairs between the 7,934 compounds in the Drug Repurposing Hub and the 10 antibiotics listed above and

removed any pairs whose adjuvant was present in the Kulesa *et al.* [25] dataset. The final drug repurposing dataset contains 41,541 antibiotic-adjuvant pairs which we can screen using SynerGNN.

3.2 Computational Methods

To accomplish our task of building a deep learning tool to predict antibiotic synergy, we built SynerGNN, a graph neural network with several architectural changes from the description in Chapter 2 that follow contemporary methods in the field of graph neural networks.

3.2.1 Graph Attention Networks

During the aggregation step of a basic graph neural network, each node aggregates the feature vectors of its neighbors by taking the mean, sum, max, or some other combination of the feature values of each neighboring vector. Regardless of the aggregation method used, all neighbors of a given node are weighted equally when aggregated. In the context of this thesis, this may hamper the model's ability to learn molecular property prediction tasks, particularly when certain atoms or functional groups in a molecule are more important for certain properties. Instead, we wish for the model to able to determine the relative importance of all nodes in a graph. We can do so using attention.

Originally developed for sequence-based tasks such as textual understanding and reasoning [29], attention mechanisms allows graph neural networks to dynamically weigh the importance of different nodes and edges during aggregation, enabling them to focus on the most relevant information for a given task. As a result, attention-based graph neural networks have achieved state-of-the-art performance in various tasks, highlighting the importance of attention in enhancing the predictive power of graph neural networks.

We chose to implement a graph attention mechanism inspired by Veličković *et*

al. [30]. During the aggregation step, we replace the adjacency matrix used in the aggregation function with a weighted adjacency matrix $A_w \in R^{N_n \times N_n}$ created by multiplying the original adjacency matrix with a weight matrix $W \in R^{N_n \times N_n}$: $A_w = A \odot W$. For example, sum aggregation with attention is computed: $X_{\text{agg}} = A_w X$. Where previously the adjacency matrix could contain only the values 0 (indicating that two nodes are not connected) and 1 (indicating that two nodes are connected), the weighted adjacency matrix can now contain arbitrary numerical values denoting the strength of a connection between two nodes.

The weight values are computed by an attention layer that takes as input a pair of feature vectors for two nodes and outputs the attention weight between them: $f_{attn} : R^{N_f} \times R^{N_f} \rightarrow R$. This attention weight is computed for all possible pairs of nodes in a graph, even those that are not direct neighbors. The weights are then collected into the weight matrix W, which is multiplied element-wise with the attention matrix A as described above. The attention layer f_{attn} is implemented as a single-layer basic neural network, and the rest of the GNN process is left unchanged as it has been described in sections prior.

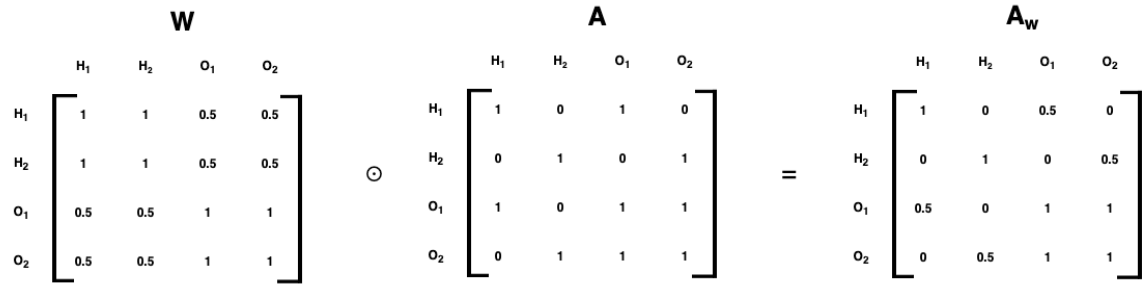


Figure 3.1: Graph attention example showing a hypothetical weight matrix for hydrogen peroxide being multiplied with the adjacency matrix to create a weighted adjacency matrix.

Figure 3.1 shows an example with a hypothetical weight matrix for hydrogen peroxide being multiplied with its adjacency matrix to create the weighted adjacency matrix for aggregation. In this example, the model has determined that the feature

vectors of neighboring nodes of different elements are less important than the feature vectors of nodes of the same element (for example, when aggregating the feature vectors for one of the oxygen atoms, the hydrogen atom node that that oxygen atom is connected to is less important the other oxygen it is connected to, hence the attention weight of 0.5 for the connections between O_1 and H_1 and O_2 and H_2).

3.2.2 Co-attention Networks

Implementing graph attention can help improve the model's intra-graph understanding of a molecule, as every atom within a molecule will learn the relative importance of its neighbors during aggregation. This can improve performance in single-molecule prediction tasks (such as predicting the lipophilicty of a single molecule). However, it would be beneficial for paired-molecule prediction tasks (such as predicting antibiotic synergy between two drugs) to also improve the inter-graph understanding of multiple molecules.

As Figure 2.9 illustrates, a basic GNN does not synthesize its understanding of separate molecules in a multi-molecule setting until the final readout step where it concatenates the individual learned latent representations. We can help the model synthesize inter-graph understandings earlier during the aggregation-transform-update process by using co-attention [31].

Where otherwise both molecules in a paired-molecule prediction task are treated independently until the final readout step, co-attention allows the model to process both molecules in context of one another from the beginning of the GNN process. To do so, we combine the adjacency matrices and atom feature matrices of the two molecules we with to predict synergy for (or some other property) into a joint adjacency matrix and joint atom feature matrix.

To create the atom feature matrix, we stack two given atom feature matrices $X_1 \in R^{N_f \times N_1}$ and $X_2 \in R^{N_f \times N_2}$ row-wise (where N_1 and N_2 are the number of atoms in molecule 1 and 2, respectively, and N_f is the number of feature in the atom feature

matrix), creating the joint matrix $X \in R^{N_f \times (N_1+N_2)}$.

The adjacency matrix is then built as follows:

1. Given two adjacency matrices $A_1 \in R^{N_1 \times N_1}$ and $A_2 \in R^{N_2 \times N_2}$, we create a new adjacency matrix $A \in R^{(N_1+N_2)\times(N_1+N_2)}$.

2. The values of A_1 are placed in the upper-left corner of A and the values of A_2 are placed in the lower-right corner of A. In other words, $A_{i,j} = A_{1_{i,j}}$ for all $i \in [0, N_1),\ j \in [0, N_1)$ and $A_{m,n} = A_{2_{m,n}}$ for all $m \in [N_1, N_1 + N_2),\ n \in [N_1, N_1 + N_2)$.

3. The remaining values in the upper-right corner and lower-left corners of A are filled with a value $\alpha \in (0, 1)$, where α is a user-specified hyperparameter that controls the strength of the co-attention connection between the two molecules. In other words, $A_{k,l} = \alpha$ for all $k \in [0, N_1),\ l \in [N_1, N_1 + N_2)$ and $A_{f,g} = \alpha$ for all $f \in [N_1, N_1 + N_2),\ g \in [0, N_1)$.

Conceptually, setting the values in upper-right and lower-left corners of A to a non-zero value introduces connections between all of the atoms of the first and second molecule. These connections are treated exactly as if there existed bonds from all of the atoms of the first molecule to all of the atoms of the second, and vice-versa, with the exception that α scales the values of the inter-graph connections. In other words, during the aggregation step of the GNN process, the inter-graph atom feature vectors (i.e. those being aggregated from the second molecule) are scaled by a factor of α.

Examples of the joint adjacency matrix and joint feature matrix are shown in Figures 3.2 and 3.3, respectively.

3.3 Training and Optimization

SynerGNN was implemented using Python and PyTorch and was trained on Brown University's high performance computing cluster, Oscar. Using both random and

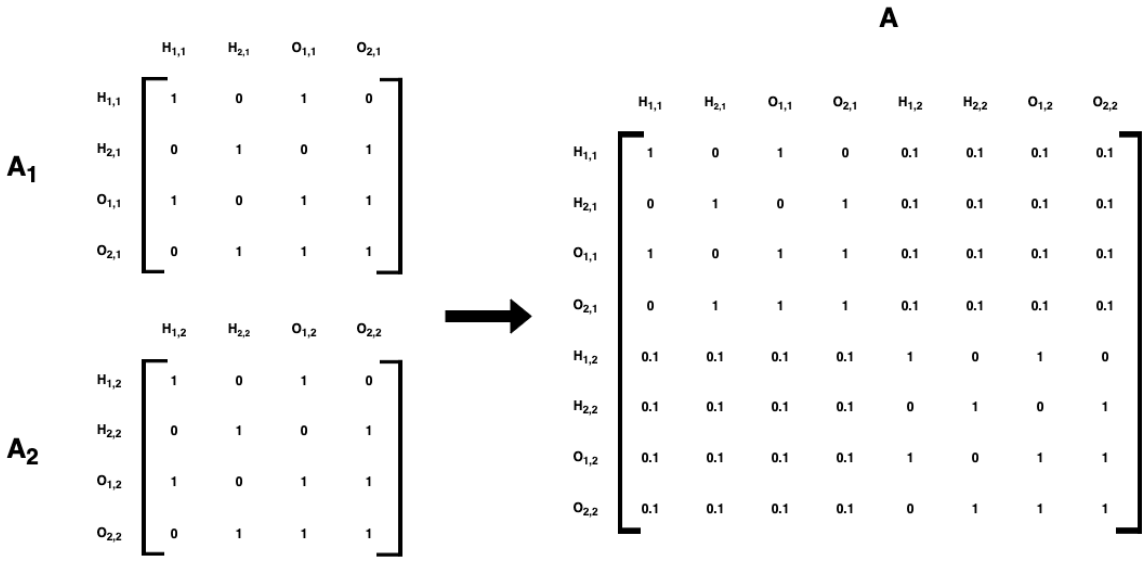


Figure 3.2: Example of how two adjacency matrices A_1 and A_2 are combined into the joint adjacency matrix A with $\alpha = 0.1$. The indices of the matrix denote the atom number and which molecule that atom belongs to (e.g. $H_{1,2}$ denotes the first hydrogen atom in the second molecule.

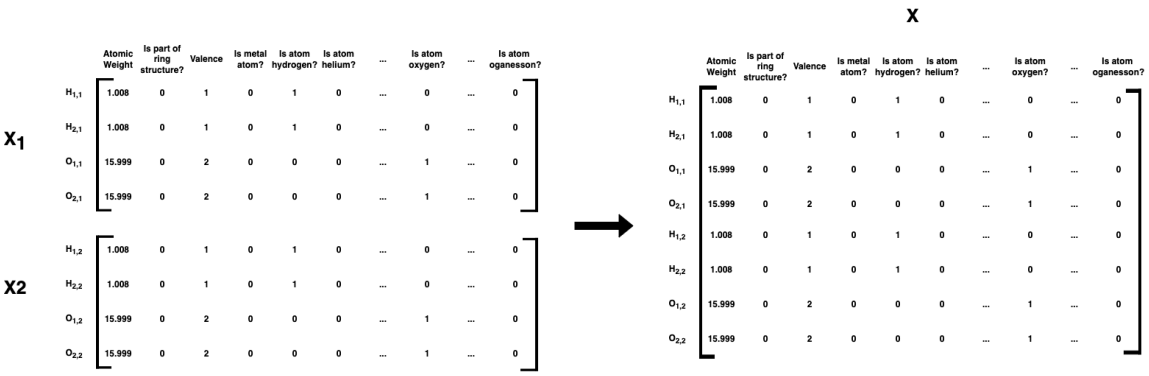


Figure 3.3: Example of how two atom feature matrices X_1 and X_2 are combined into the joint atom feature matrix.

Bayesian search techniques [32], a total of 2,722 instances of SynerGNN were trained in order to determine optimal model architectures and hyperparamters and to evaluate the performance improvements of graph attention and co-attention. Figure 3.1 summarizes the optimal hyperparameters that were determined via training.

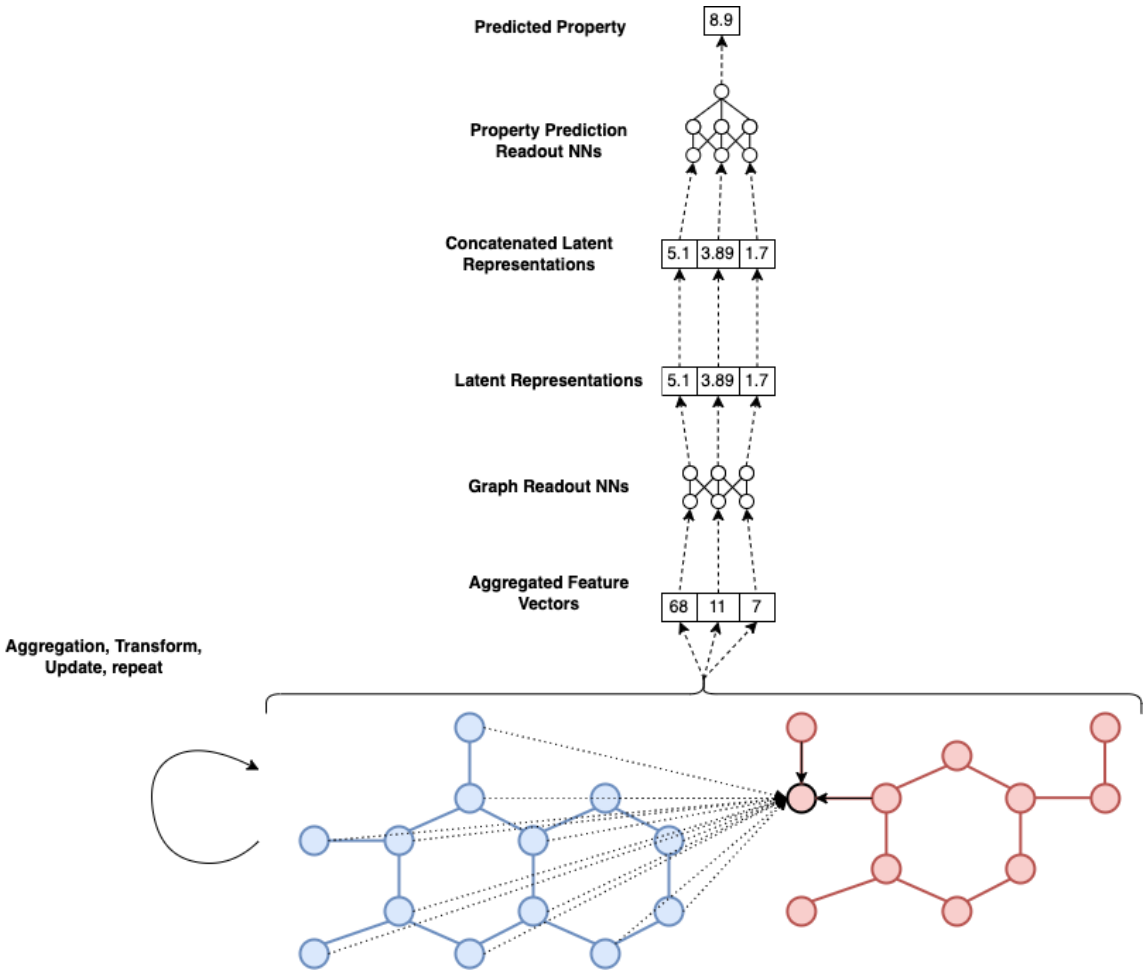


Figure 3.4: High-level overview of the GNN process with co-attention. For visual simplicity, co-attentive connections are only shown for a single centroid atom in the red molecule. In practice, every atom in both molecules has co-attentive connections to all atoms in the other molecule.

We trained several versions of SynerGNN without and without the computational methods above. The base model, referred to as SynerGNN, is a basic graph convolutional network that follows the description of a GNN put forth in Chapter 2, whereas SynerGNN-GA and SynerGNN-CoA implement the graph attention mechanism and co-attention mechanism, respectively.

3.3.1 Performance Evaluation Metrics

In evaluating the performance of SynerGNN, we primarily report the root mean squared error (RMSE, or L2 loss) and Kendall's rank correlation coefficient (Kendall's

Table 3.1: Summary of SynerGNN hyperparameters for lipophilicity and antibiotic synergy datasets

Parameter	Value for Lipophilicity	Value for Antibiotic Synergy
Activation Function	LeakyReLU [33]	LeakyReLU [33]
Aggregation Function	Sum	Sum
Depth	5	5
Dropout Probability	0.30	0.4
Epochs	100	100
Hidden Size	1024	1024
Learning Rate	0.005	0.005
Loss Function	Mean Absolute Error	Mean Absolute Error
Optimizer	Adagrad [34]	Adadelta [35]

τ).

RMSE

RMSE is a commonly-used metric to evaluate the performance of regression models which can help us understand a model's average error across a dataset and is calculated by aggregating the square of model prediction errors and subsequently taking the square root of the average of these errors:

$$\text{RMSE} = \sqrt{\frac{\sum(\text{actual} - \text{predicted})^2}{\#\text{ of datapoints}}}$$

It can be interpreted as the weighted average error between the predictions our model makes and the actual values in the dataset and allows us to evaluate the performance of our model at the same scale as the data it is being trained on.

Kendall's τ

The ability for our model to accurately predict individual values is incredibly important. However, for many tasks it is also important to ensure that the values the

model predicts are correctly ranked within the dataset. In the context of our work, it is more important that our model is able to accurately separate synergistic antibiotic pairs from non-synergistic pairs by ranking their predicted Bliss scores than it is to simply predict their Bliss scores outside of the context of the datapoints.

One method of evaluating this ranking ability is to calculate the Kendall rank correlation coefficient (or Kendall's τ). When used in evaluating machine learning models, Kendall's τ reports how well a model is able to rank its predicted values compared the actual values of the data. In other words, it checks if the "first" value in the set of actual values of a dataset (when ranked by some means such as in increasing or decreasing order) is the same as the "first" value in the set of predicted values by a model when run on that dataset and repeats this process for the "second", "third", and "fourth" datasets, and so on.

Kendall's τ is calculated as follows: Let $y_0, ..., y_i$ be a set of actual values from a dataset and $\hat{y}_0, ..., \hat{y}_i$ be the set of corresponding predicted values when a model is run on that dataset. A pair of observations $(y_i, \hat{y}_i)$ and $(y_j, \hat{y}_j)$ is concordant if both $y_i > y_j$ and $\hat{y}_i > \hat{y}_j$ or $y_i < y_j$ and $\hat{y}_i < \hat{y}_j$. Ties are ignored, and the final value of τ is

$$\begin{aligned}\tau &= \frac{(\text{number of concordant pairs}) - (\text{number of discordant pairs})}{(\text{number of pairs})} \\ &= 1 - \frac{2(\text{number of discordant pairs})}{\binom{n}{2}}\end{aligned}$$

A τ of 1 indicates that the ranks of the predicted values perfectly match the ranks of the actual values, whereas the value of -1 indicates that the ranks of the predicted values are fully different than the ranks of the actual values. A τ of 0 indicates that there is no correlation between the predicted and actual values, suggesting that a model is randomly predicting values.

Chapter 4

Results

In this section, we present a summary of SynerGNN's performance on several tasks. While we will discuss the performance of the model in predicting the lipophilicty values of single molecules in the context of the benchmark lipophility dataset, the majority of this analysis will focus on discussing the model's performance in predicting antibiotic synergy between multiple molecules.

The implications of the results mentioned here are discussed in Chapter 5.

4.1 Single Molecule Lipophilicity Data

To fairly evaluate SynerGNN's ability to understand molecule structures, we first trained the model on the single-molecule lipophilicity data discussed in Section 3.1.2. Using the model hyperparameters above, we trained SynerGNN across a spectrum of random initial weights and recorded the performance metrics of the best-performing model. We also evaluated the performance of SynerGNN-GA on this dataset.

Table 4.1 summarizes the performance metrics of SynerGNN and SynerGNN-GA on the single-molecule lipophilicity data. Across the held-out validation data, we found that the base implementation of SynerGNN outperformed the implementation of SynerGNN with graph attention.

It also important to contextualize this performance with other published molecular property prediction models. A comparison between SynerGNN/SynerGNN-GA and

Table 4.1: Summary of SynerGNN performance on single molecule lipophilicity dataset

	Training Data		Validation Data	
Model	RMSE	Kendall's τ	RMSE	Kendall's τ
SynerGNN-GA	0.374	0.809	0.619	0.668
SynerGNN	**0.373**	**0.807**	**0.598**	**0.680**

Table 4.2: Comparison of SynerGNN performance on single model lipophilicity dataset to existing models

Rank	Model Name	Year	RMSE
1	MOLFORMER-XL [36]	2023	0.528
2	A-FP [37]	2019	0.578
3	**SynerGNN**	**2023**	**0.598**
4	**SynerGNN-GA**	**2023**	**0.619**
5	C-SGEN+ Fingerprint [38]	2019	0.650
6	GC [27]	2015	0.655
7	D-MPNN [27]	2019	0.683
8	Weave [27]	2016	0.715

other published models is shown in Table 4.2. The table only reports RMSE as Kendall's τ was not published for the other models.

Because co-attention requires multiple molecules per datapoint in a dataset to provide any performance improvement, SynerGNN-CoA was not evaluated on the single-molecule lipophilicity dataset.

4.2 Multi Molecule Antibiotic Combination Data

After validating SynerGNN's performance on single-molecule tasks, we next evaluated SynerGNN on the task at the heart of this thesis — predicting antibiotic synergy of pairs of molecules.

Table 4.3 summarizes the performance of the base SynerGNN implementation,

Table 4.3: Summary of SynerGNN performance on multi-molecule antibiotic synergy dataset

	Training Data		Validation Data	
Model	RMSE	Kendall's τ	RMSE	Kendall's τ
SynerGNN-GA	0.093	0.690	0.140	0.336
SynerGNN-CoA	**0.092**	**0.644**	**0.135**	**0.352**
SynerGNN	0.099	0.630	0.141	0.342

Table 4.4: Comparison of SynerGNN performance performance on multi-molecule antibiotic synergy dataset to existing model

Rank	Model Name	Year	RMSE	Kendall's τ
1	**SynerGNN-CoA**	**2023**	**0.135**	**0.352**
2	Chemprop [39]	2019	0.137	0.312
3	**SynerGNN**	**2023**	**0.141**	**0.342**
4	**SynerGNN-GA**	**2023**	**0.140**	**0.336**

SynerGNN with graph attention, and SynerGNN with co-attention on the dataset discussed in Section 3.1.1.

In this setting, SynerGNN was compared with the performance of Chemprop [39], another deep learning model capable of performing multi-molecular property prediction tasks.

4.3 Drug Repurposing Data

In order to predict novel synergistic antibiotic combinations, we ran SynerGNN on the 41,541 antibiotic-adjuvant pairs described in Section 3.1.3.

The top 20 antibiotic-adjuvant pairs ranked by Bliss score as predicted by SynerGNN are reported in Table 4.5.

Table 4.5: Top 20 antibiotic-adjuvant pairs sourced from the drug repurposing data, ranked by predicted Bliss score

Adjuvant	Antibiotic	Predicted Bliss Score
mometasone-furoate	ampicillin	0.443
CaMKII-IN-1	chloramphenicol	0.436
triamterene	cycloserine	0.433
triamterene	norfloxacin	0.426
phenethicillin	fosfomycin	0.416
UNC2025	chloramphenicol	0.414
amcasertib	chloramphenicol	0.409
entrectinib	chloramphenicol	0.405
derazantinib	cycloserine	0.404
CaMKII-IN-1	cycloserine	0.398
mometasone-furoate	chloramphenicol	0.396
PSI-697	chloramphenicol	0.395
derazantinib	chloramphenicol	0.391
BIO-1211	chloramphenicol	0.387
CaMKII-IN-1	erythromycin	0.383
6-aminopenicillanic-acid	fosfomycin	0.378
BIO-1211	tetracycline	0.377
CPI-169	ampicillin	0.375
mometasone-furoate	cycloserine	0.374
FIPI	ampicillin	0.373

Chapter 5

Discussion

5.1 Key Findings

Data presented in the previous section suggests that graph neural networks such as SynerGNN can be accurate predictors of both physiochemical properties and biochemical activity of compounds and are capable of doing use by studying and understanding their underlying molecular structures. In particular, we trained and evaluated SynerGNN on two learning problems — prediction of a molecule's lipophilicity and prediction of two molecules' combined antibiotic synergy. The first task was intended to both benchmark SynerGNN against other contemporary single-molecule prediction models and to validate that it is capable of learning and understanding single molecule inputs before expanding the task domain to encompass multiple molecules. The second task models the goal we originally set out to achieve in this thesis — discovering new synergistic antibiotic combinations using a fast, scalable computational model trained on the molecular structures of known synergistic pairs.

5.1.1 Single Molecule Lipophilicity Data

Establishing that a model performs well in a similar but distinct problem domain is an important step in verifying that the model is generalizable in its ability to understand its inputs rather than spuriously over fitting input data to its intended goal. In evaluating the results of the lipophilicity prediction task, we show SynerGNN per-

forms well in predicting LogD values of molecules in Table 4.1. In particular, we can interpret the model's validation RMSE as saying that, on average, the model is able to predict the lipophilicity of a molecule within ± 0.598 LogD of the true value, or within half a standard deviation of the data. We can also interpret the model's validation Kendall's τ. If the model was incapable of correctly ranking any its predicted lipophilicity values, we would expect to see a Kendall's τ near or below 0. However, a Kendall's τ of 0.680 means that the model is able to favorably rank predicted lipophilicity values, thus suggesting that the model, even if it cannot predict LogD with an extremely high degree of accuracy, does a good job of distinguishing highly lipophilitic molecules from moderately lipophilicity molecules, moderately lipophilicity molecules from poorly lipophilicity molecules, and so on.

We can also validate SynerGNN's performance by comparing it to other contemporary single-molecule prediction models. Table 4.2 compares the validation RMSE of SynerGNN and SynerGNN-GA to similar models, and we can see that SynerGNN performs third-best among its peers behind the Attentive-FP model by Xiong *et al.* [37] and the MOLFORMER-XL model by Ross *et al.* [36]. These results suggest that SynerGNN is sufficiently capable of learning to complete general molecular property prediction tasks given molecular structures, and although we do not see state of the performance performance of SynerGNN in the domain of single-molecule prediction tasks, these results were enough to validate the underlying architecture of SynerGNN before we applied the model to the more complex and less-well-studied domain of multi-molecule prediction tasks.

Performance Impact of Graph Attention

Interesting, our results showed that inclusion of the graph attention mechanism (SynerGNN-GA) decreased validation data performance on this task, which might suggest that equal weighing of a node's neighboring feature vectors is superior to weighting them independently. However, if this were the case, we would expect

SynerGNN-GA to learn to weigh neighbors equally, which is not the case. What is far more likely is that the graph attention mechanism introduces a degree of over-fitting to the model. We can see evidence of this by comparing the training data RMSE and Kendall's τ of SynerGNN-GA to SynerGNN. Both models perform similarly on the training data, but the base SynerGNN implementation performs better on the validation data. For SynerGNN-GA, stronger performance on the training dataset and weaker performance on the validation dataset suggests that the neighbor-weighting schemes learned by the graph attention mechanism do not generalize well to the validation data.

Comparison of SynerGNN to Better-Performing Models

To better understand how SynerGNN can be improved in future work, we compare its architecture to the architectures of the models that outperform it in the single-molecule domain.

We first compare SynerGNN to MOLFORMER-XL, the top performing model on our lipophilicity benchmark dataset. Unlike most other models we compared SynerGNN to, MOLFORMER-XL uses sequence-based transformers to learn on string representations of molecules (known as SMILES strings) rather then graphs. SMILES (and its derivatives SMARTS and SELFIES) strings were generally thought to lack sufficient topological and structural representative power to be of use for property prediction models, but, using an attention mechanism that encodes inter-string importance of characters and tokens, the authors were able to attain state of the art performance on many property prediction tasks and show that MOLFORMER-XL is capable of learning the spatial relationships between atoms in a molecule that were previously thought to be representable only with graphs. In particular, MOLFORMER-XL benefited from an extensive pre-training process on a total of 1.1 billion unlabeled, task-agnostic molecules to help the model learn effective latent representations (much in the same way that reading non-fiction literature may build one's vocabulary and

prose to help them write future work). Such pre-training required over 3,300 GPU-hours to complete, compared to the 8 equivalent GPU-hours that were used to train an instance of SynerGNN. We believe that this pre-training process is a key reason why MOLFORMER-XL outperforms SynerGNN on the benchmark task and that SynerGNN would similarly benefit from pretraining should the computational resources to do so become available.

We next compare SynerGNN to the Attentive-FP model by Xiong *et al.* [37]. Like SynerGNN, Attentive-FP represents molecules as graph structures and utilizes a graph attention mechanism to better learn task-relevant substructures within a molecule. However, their implementation of attention and of the model's graph neural network differs from SynerGNN in two key ways. First, whereas the attention mechanism of SynerGNN simply multiplies some weight matrix W with the adjacency matrix of a molecule A to weight the intragraph node connections, the attention mechanism of Attentive-FP twice normalizes the output of this multiplication step. In short, the goal of normalization is to increase model generalizability by reducing the variability of a model's latent representations. In the case of Attentive-FP, twice normalizing the output of attention step may help reduce the variability of attention weights. This variability may be a contributing factor to the overfitting seen introduced by SynerGNN-GA, and more work should be devoted to understanding the variability of SynerGNN-GA's attention weights to understand if normalization could fix its tendency to overfit. Second, Attentive-FP uses gated recurrent units (GRUs) to aggregate and update node latent representations rather than the simple matrix multiplication and fully-connected neural networks used by SynerGNN. GRUs are commonly utilized in sequence-based neural networks to help a model remember important information in a time- or sequence-based manner (such as remembering the subject in a lengthy sentence in a text-classification task). The use of GRUs may help Attentive-FP better retain information in each atoms' latent representation, whereas successive passes of the aggregation-transform-update steps of SynerGNN may cause

the latent representations of each node in a molecule to become dilute of information, especially at higher depths of the model.

Comparison of SynerGNN to Worse-Performing Models

Likewise, to understand which components of the SynerGNN architecture have positively contributed to performance in the single-molecule domain, we can compare our model to the architectures of worse-performing models.

Like Attentive-FP, C-SGEN differs from SynerGNN in how it aggregates and updates nodes' latent representations. C-SGEN uses a convolutional neural network to pass convolutional kernels over stacks of neighboring atom's feature vectors:

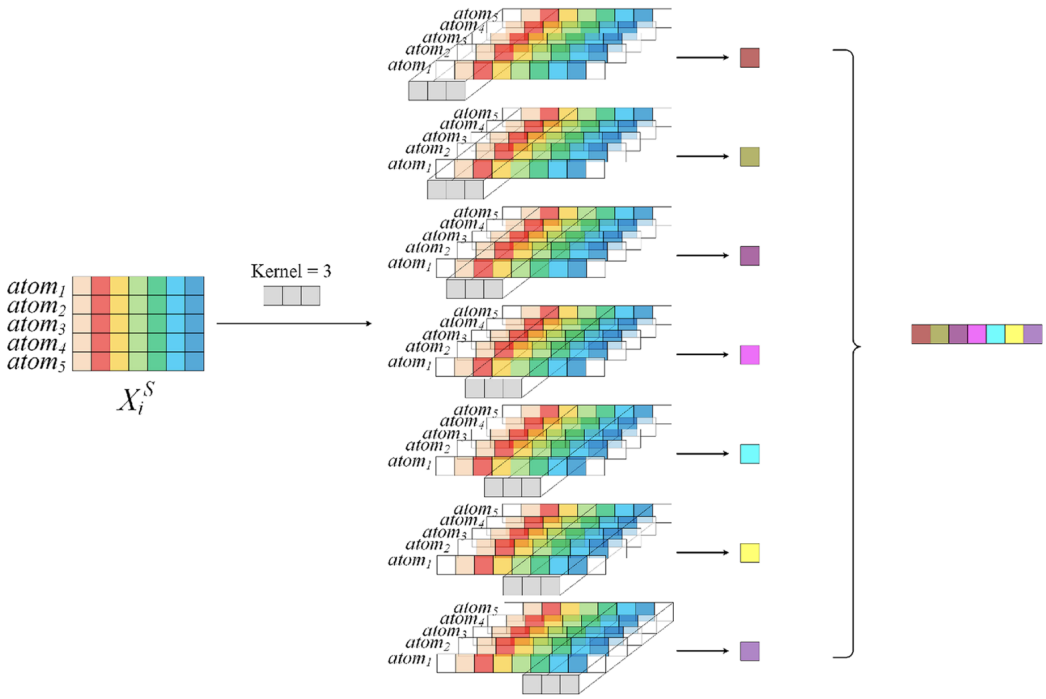


Figure 5.1: The convolutional mechanism that C-SGEN uses to aggregate neighboring molecules' feature vectors [38].

Where $atom_1$, $atom_2$, ..., $atom_5$ are the feature vectors of $atom_1$ and its neighbors and X_i^S is a matrix built by stacking these atom feature vectors. A convolutional kernel of size 3 is passed over the stacked features, aggregating them into a new atom feature vector for $atom_1$. Like SynerGNN, successive passes of this process can build

up multiorder neighborhood latent representations for each atom. While convolutional neural networks (CNNs) perform well on structured data (such as molecules, images, or time-sequence data), the way they are being applied in C-SGEN does not take advantage of so. Normally, the kernels of a CNN are passed across the data such that some substructure of the data will be captured within the kernel (such as capturing a neighborhood of atoms in a molecule, a small subsection of an image, or a several-month timeframe within a years-long dataset). In C-SGEN, the kernel is instead being passed across the features of every atom in a neighborhoods' feature vector. There is no structural information or sense of locality encoded or enforced by the ordering of the features in these vectors. In other words, we are free to reorder the features in the vectors, and doing so would not break or fundamentally change how a given molecule is represented (as opposed to a change that would break a molecule's representation such as changing the order of molecules in the graph) but would change how the kernels perceive neighboring features. We suppose that this sensitivity to atom feature order is a reason why C-SGEN underperforms SynerGNN. Perhaps applying the kernels to operate across the atoms in a neighborhood rather than across the features in each atoms' feature vector would improve performance on this task, but further studies of so are beyond the scope of this thesis.

The 6th-best performing model, GC, is a baseline graph convolutional neural network similar to SynerGNN. Several design decisions of the GC model's convolutional architecture more closely follow traditional CNNs. For example, their aggregation method of max-pooling (following the max-pooling layers seen in traditional CNNs) takes the maximum of each feature for a given set of node feature vectors, while the best-performing SynerGNN models take the sum of all feature vectors. The GC model also lacks dropout layers, the use of which has since become standard in deep learning models to reduce overfitting and improve model generalizability. Unfortunately, exact architectural information about the GC model is rather sparse in its source paper [40] The authors do not report the specific hyperparameters that were used to build

their GCNN such as depth or hidden layer size, thus we can only speculate that the existence of dropout layers, differences in hyperparameters, sub-architectual changes in the exact code implementation of SynerGNN, or some combination of the above, is why SynerGNN outperforms GC.

Finally, we compare SynerGNN to the weave model implemented in [27]. Like the GC model, architectural specifics are unfortunately not mentioned in the paper. However, we can broadly compare how weave models work with SynerGNN to elucidate why SynerGNN outperforms them. Like SynerGNN and other GCNNs, weave models work directly with graph-structured molecules. Unlike SynerGNN and other GCNNs, however, weave models update the feature vectors of atoms in the molecule using pairs of every single atom in the molecule, even those that are not directly bonded to each other. Then, separate atom-pair features are created for all directly bonded atoms by aggregating the features vectors of both atoms in the pair. These feature vectors are then used downstream in the model's readout layer. Because atom feature vectors are updated using all possible pairs of atoms, we speculate that weave models lose a substantial amount of structural representative power compared to GCNNs, and that this structural representative power is particularly important for molecular property prediction tasks.

5.1.2 Multi Molecule Antibiotic Synergy Data

After validating that SynerGNN is capable of performing single molecule prediction tasks, we trained SynerGNN on the multi-molecule antibiotic synergy dataset that is the focus of this thesis. Results on this dataset are shown in Table 4.3.

While SynerGNN's training RMSE and Kendall's τ are suggestive of favorable performance on this task, the validation RMSE and Kendall's τ indicate otherwise. The validation RMSE being close to the standard deviation of the antibiotic synergy dataset (0.162) and the Kendall's τ that is closer to 0 than not suggests that SynerGNN is learning to naively predict that the antibiotic synergies of all datapoints in

the antibiotic synergy dataset are centered around the mean, even those that should not be such as highly synergistic or antagonistic combinations. We can confirm this by visually analyzing the distributions of the Bliss scores of the antibiotic dataset and those of the predicted Bliss scores by the model. Figures 5.2 and 5.3 show the distribution of Bliss scores in the antibiotic synergy dataset and those predicted by SynerGNN, respectively.

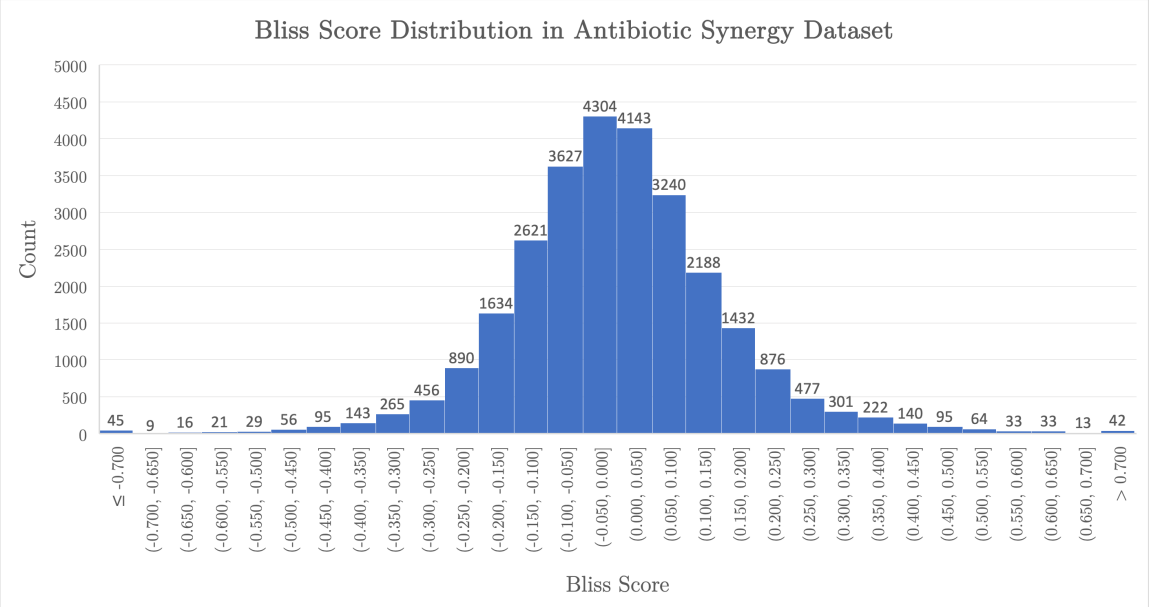


Figure 5.2: Bliss score distribution of the antibiotic synergy dataset.

In both histograms, we care most about the values in the tails, which correspond to the combinations that are either highly synergistic or highly antagonistic. For visual brevity, we group the values in the tails into buckets with Bliss score greater than 0.7 (which is the cutoff value used by Kulesa *et al.* [25] to determine synergistic combinations vs. non-synergistic combinations) or less than -0.7. Combinations with Bliss scores in between these values represent indifferent combinations — those with no emergent positive or negative antibiotic killing activity. In Figure 5.2, we can see that the dataset contains a total of 41 synergistic (Bliss score $\geq$ 0.7) and 45 antagonistic (Bliss score $\leq$ -0.7) combinations. However, when we examine the Bliss score distributions of the SynerGNN-generated predictions in Figure 5.3, we can see

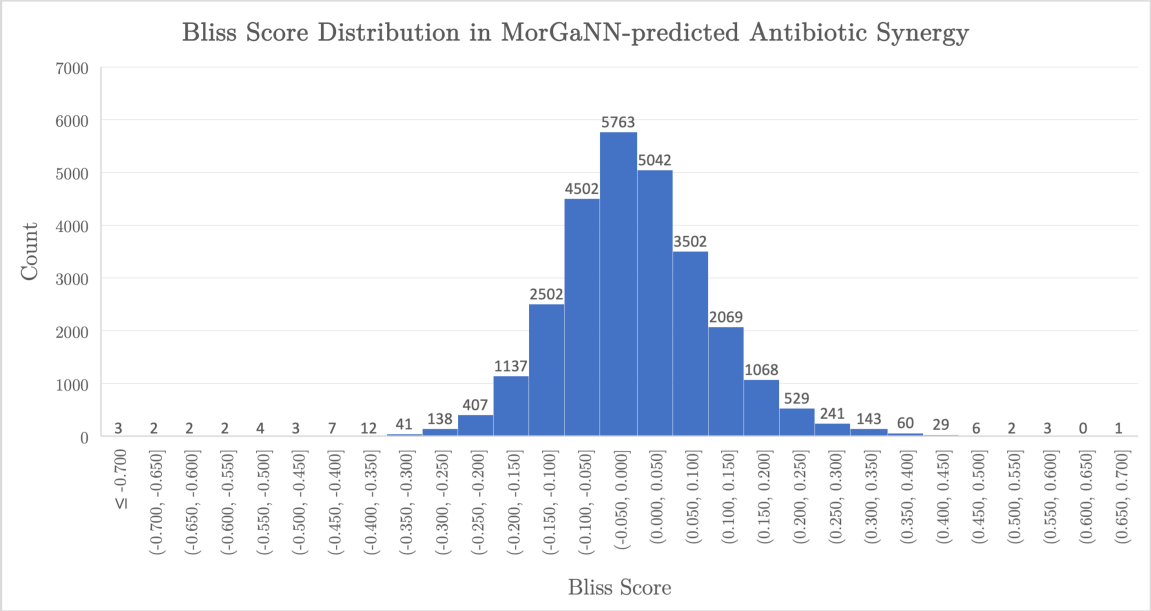


Figure 5.3: Bliss score distribution of SynerGNN-predicted antibiotic synergies.

that the the model fails to predict that any of the combinations in the dataset are synergistic and identifies only three of the antagonistic combinations.

Analyzing the distributions of the Bliss scores in the training dataset elucidates why SynerGNN struggles to accurately predict the values we deem important. The antibiotic synergy data we are working with is an example of a highly imbalanced dataset — one in which the distribution of the target variable is highly skewed towards a small range of values, and where there are relatively few instances of other target variable values (particularly ones of interest). In our case, we can see that the dataset is clearly skewed towards indifferent combinations, as only 0.16% of all datapoints in the dataset represent synergistic combinations.

Models like SynerGNN struggle with highly imbalanced data because they may not see enough examples of datapoints from regions outside the peak of the distribution to learn how to predict their values accurately. Datapoints from these regions may be underrepresented in the training set, making it difficult for the model to learn patterns that are specific to these values. When a regression model is trained on imbalanced data, it is likely to prioritize predicting datapoints from the peak of the distribution,

as it has more examples of these to learn from. As a result, the model may perform well in predicting these datapoints, but it may not be able to generalize well to others. In the context of our data, our model is not exposed to enough examples of synergistic antibiotic combinations to learn the structural and molecular patterns that separate synergistic combinations from asynergistic combinations.

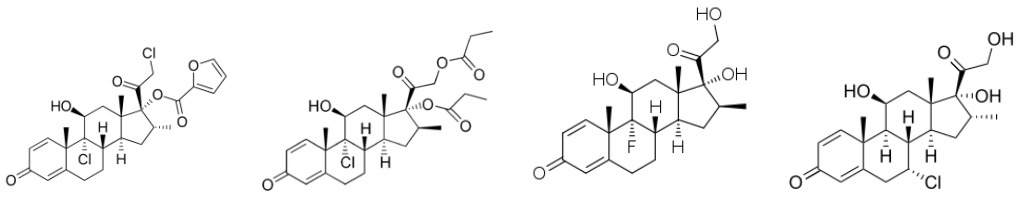
5.1.3 Drug Repurposing Data

Despite what we established in the analysis of SynerGNN's performance in predicting antibiotic synergy, some insight and model validation can be gleaned by analyzing what the model considers to be the most synergistic of antibiotic combinations (those listed in Table 4.5) in the drug repurposing dataset. It is worth reiterating that none of the combinations within this dataset were present in the training or validation data, thus making the analysis of these predictions a true test of validation for the model.

Although the predictions in this dataset are characterized by low overall Bliss scores (none of the combinations are predicted to have a Bliss score over 0.5), there is some merit in analyzing the predictions. After all, even if the model has incorrectly believes that all Bliss scores fall into a very narrow range, it has likely still learned to identify that some combinations have higher Bliss scores than others. Thus we can disregard the Bliss scores predicted by the model and instead simply consider the rank ordering of the combinations from most-synergistic to least-synergistic.

The combination with the highest Bliss score as predicted by SynerGNN is mometasone-furoate and ampicillin. Mometasone-furoate is a topical or inhaled methasone-type glucocorticoid medication primarily used to treat skin conditions, hay fever, and asthma [41]. Studies by Neher *et al.* [42] showed that mometasone-furoate does indeed poses innate antibacterial properties in being able to kill *Streptococcus milleri* and *Streptococcus pyogenes*. Interesting, no killing ability was detected for *E. coli* in their study, despite *E. coli* being the subject of the antibiotic killing assays used

by Kulesa *et al.* [25] in the antibiotic synergy training data for SynerGNN. However, other methasone-type glucocorticoid compounds have shown promise as adjuvants in antibacterial combinations. Although in-vitro studies are lacking, Pitcairn *et al.* [43] showed that a combination of the glucocorticoid dexamethasone and ampicillin maximized survival rates of *E. coli*-injected rats in-vivo over ampicillin alone. The antibiotic synergy training data does contain several entries of methasone-type glucocorticoid-ampicillin combinations, namely beclomethasone-ampicillin, alclometasone-ampicillin, and betamethasone-ampicillin. Of those, only beclomethasone-ampicillin was found to exhibit some degree of antibiotic synergy with ampicillin (Bliss score 0.568 compared to 0.011 and 0.015 of the others). Structural analysis of these glucocorticoids shows that beclomethasone shares a highly conserved scaffold with mometasone versus alclometasone or betamethasone, suggesting that SynerGNN has learned to associate these conserved structural features with ampicillin antibiotic synergy (see Figure 5.4.



(a) (b) (c) (d)

Figure 5.4: Chemical structures of mometasone (a), beclomethasone (b), betamethasone (c), and alclometasone (d).

The next-most synergistic combination is CaMKII-IN-1, a highly selective Calcium–calmodulin (CaM)-dependent protein kinase II (CaMKII) inhibitor [44], and chloramphenicol. CaMKII-IN-1 has not been studied in-vitro to discern any antibiotic or antibiotic adjuvant activity, although studies by Chi *et al.* [45] show that CaMKII inhibiton reduces *E. coli* invasion of human brain microvascular endothelial cells. CaMKII-IN-1 is highly-represented in SynerGNN's top-predicted combinations, also appearing in the 10th- and 15th-most synergistic combinations paired with cycloserine

and erythromycin, respectively. However, chemical similarity searches using a variety of molecular fingerprinting methods failed to find a reasonably-similar molecule in the training data, so it remains unknown why SynerGNN deems CaMKII-IN-1 to be a highly-synergistic adjuvant.

Triamterene, a potassium-sparing diueretic, is similarly highly-represented in SynerGNN predictions, appearing in the third- and fourth-most synergistic combinations. Triamterene, and other drugs in its class, elicit diuretic activity by blocking the epithelial sodium channel (ENaC) which is primarily expressed in human kidneys. While triamterene has not traditionally been studied as an antimicrobial agent for *E. coli*, a study conducted by Giunta *et al.* [46] showed that both triamterene and amiloride, a similar potassium-sparing diueretic, exhibit *Streptococcus faecalis* growth-inhibitory-activity, and Sugiyama *et al.* [47] showed that amiloride inhibits bacterial flegellar motor activity interrupting cellular motility of alkalophilic *Bacillus*. Islam *et al.* [48] further showed that several amiloride analogs were capable of reducing motility in *E. coli*. In their original work, Giunta *et al.* [46] speculate that potassium-sparing diueretics interfere with bacterial cell Na+ balance, ultimately leading to cell death. While cell motility is critical for bacterial virulence and survival in-vivo, it alone cannot confer growth-inhibitory activity in-vitro. However, Islam *et al.* [48] showed that several amiloride analogs were capable of significantly inhibiting *E. coli* growth but speculate that it is unrelated to the compounds' antimotility effects. The specific mechanism of action (MOA) for this growth-inhibitory effect is unknown, although it is reasonable to believe that the MOA that gave rise to the growth-inhibitory results in Islam *et al.* [48]'s and Giunta *et al.* [46]'s results are the same. A structural similarity search through the antibiotic synergy training data revealed TG100713, a pan-PI3K inhibitor that has not been directly studied as an antibiotic [49], that is highly similar to triamterine (see Figure 5.5) and was shown to exhibit generally synergistic activity with cycloserine. It is likely that SynerGNN identified triamterene as being structurally similar to TG100713, thus sharing its synergistic behaviour. Al-

though amiloride is present in the antibiotic synergy training data, its recorded Bliss scores generally indicate indifference.

(a) (b)

Figure 5.5: Chemical structures of triamterene (a) and TG100713 (b).

Overall, these results indicate that SynerGNN is capable of learning common structural features of antibiotic adjuvants and identifying other molecules that contain these structures.

5.2 Limitations and Future Work

As was stressed in Section 5.1.2, the key limitation in this work is that of data. Namely, we lack enough positive synergistic signals within the dataset, contributing to the extreme imbalance between the counts of synergistic and asynergistic pairs in the dataset. This extreme of a class imbalance within a model's training data poses a major challenge to generalizability and accuracy, as can be seen in the SynerGNN-predicted Bliss score distributions. While recent computational methods can improve regression model training performance on highly imbalanced datasets [50], there will always exist a limit to the amount of structural-synergy information that can be extracted from datasets that lack large amounts of positive synergistic pairs. Establishing more a complete dataset of antibiotic synergy will be critical for the successful continued application of machine learning techniques to this space. Thus, future work should be jointly devoted to implementing these computational methods and to building a larger dataset of positive synergistic combinations through literature searches and continued high-throughput screening.

Another limitation of the SynerGNN model is lack of interpretability. While we

have shown that SynerGNN is capable of using the molecular structures of known synergistic antibiotic adjuvants to predict novel potential adjuvants, it is not abundantly clear what structural features the model uses in predicting synergy. We can investigate the underlying training data to find structural matches for adjuvants that SynerGNN predicts to be highly synergistic, but this technique fails for adjuvants like CaMKII-IN-1 that lack clear analogs in the training data. From the the perspective of a medicinal chemist, it is critical that we extract the chemical scaffolds, motifs, and moieties that SynerGNN deems as indicators of antibiotic synergy to serve as the starting point for continued screening and fragment-based drug design efforts and for mechanism of action elucidation. While graph attention mechanisms introduce a means to interpret our model by producing intuitive weight values, the attention mechanisms implemented in this work are limited to weighting the connections of neighboring pairs of atoms in a molecule. Rarely do single atom pairs give rise to biochemical activity such as microbial growth inhibition. Rather, these effects are driven by larger molecular substructures such as rings and functional groups that provide binding surfaces and binding potential to allow these adjuvants to interact with bacterial proteins. For full structural interpretability, we thus desire to implement methods to generate attentional weights that apply to the molecular structures that exist within our potential adjuvants instead of just atom pairs. Alternatively, Gurvic *et al.* [51] use matched molecular-pair analysis to identify small structural differences between pairs of molecules that lead to the emergence of antibiotic activity in gram-negative bacteria. Running similar analysis on the results of SynerGNN may help us elucidate the substructures that SynerGNN has learned to associate with synergy.

We also recognize the limitations of naively using the drug repurposing library described in Section 3.1.3. Repurposed drugs, by definition, will have non-antibiotic physiological effects in patients if administered as adjuvants. Many of these effects may be unwanted or dangerous to patient health, particularly when co-administration

with an antibiotic may lead to as-yet-unknown drug-drug adverse events. In a real drug discovery setting, it will be critical to build a repurposing library of maximally tolerable compounds with little potential for adverse interactions with antibiotics. This also highlights the need to consider drug concentrations in these synergistic combinations. It is reasonable to believe that some repurposable drugs may be capable of exhibiting antibiotic synergy as adjuvants at concentrations too low to produce either the physiological effect they were originally developed for or any unwanted side effects. At the same time, the growth inhibitory effects of some compounds may only emerge at concentrations intolerable for healthy human cell function. An ideal antibiotic adjuvant is thus one that not only shows strong antibiotic synergy with its partner antibiotic, but is one that shows such strong synergy at very low concentrations. SynerGNN, as it stands, does not consider the drug concentrations of either the antibiotic or the adjuvant in predicting synergy, thus future work should be devoted to altering the architecture of the model to explicitly account for drug concentration.

Lastly, we must stress that SynerGNN is not intended to serve as a replacement for the in-vitro study of potential antibiotic adjuvants. Rather, it is meant to be a tool to accelerate the process of prioritizing which compounds are studied as adjuvants by screening out true-negative compounds (the in-vitro study of which would likely be wasteful) and selecting for true-positive compounds. In-vitro studies are even more important for predicted adjuvants like CaMKII-IN-1 which lack clear analogs in the training dataset that may provide insight for their mechanism of synergistic action. Instead, these kinds of compounds may exhibit novel mechanisms of action, just as the work of Stokes *et al.* [18] predicted the antibiotic activity of the drug Halicin which was later shown to exhibit a novel mechanism of interfering with bacterial cells' ability to regulate pH balance. We thus hope that the compounds predicted to be potent adjuvants by SynerGNN will one day be studied in a wet lab environment, not only to validate SynerGNN's performance but to advance progress in the fight

against antimicrobial resistant bacteria.

Chapter 6

Conclusion

The rise of antimicrobial resistant (AMR) bacteria is one of the world's most pressing public health crises, and our fight against such AMR bacteria is hampered by the time-, labor-, and cost-intensities of discovering novel antibiotics. However, drug repurposing paradigms offer us the opportunity to forgo many of these costs by repositioning existing drugs from their original medicinal purpose into being antibiotic adjuvants, or compounds that bolster the growth-inhibitory effects of antibiotics when prescribed in combination. Through the integration of modern deep learning techniques and massive amounts of activity data produced via high-throughput screening of antibiotics and potential adjuvants, we can effectively train models to discover and predict these novel synergistic antibiotic combinations. Towards that goal, we developed and trained SynerGNN, evaluated its performance on both antibiotic and non-antibiotic activity datasets, and took strides to understand the compounds that it predicted to have the highest antibiotic synergies. We also highlighted the challenges of working with limited antibiotic synergy datasets and justified the need for a more complete dataset of antibiotic synergy data and the implementation of machine learning techniques that will help us interpret the molecular structures that SynerGNN associates with synergistic activity. Ultimately, models such as SynerGNN will serve as important tools in the continued fight against antimicrobial resistant bacteria in accelerating the screening process of potential antibiotic adjuvants and in discover-

ing novel mechanisms of synergistic action that can be exploited to completely break through existing bacterial resistance mechanisms.

Bibliography

[1] G. B. Migliori, G De Iaco, G Besozzi, R Centis, and D. M. Cirillo, "First tuberculosis cases in italy resistant to all tested drugs," *Weekly releases (1997–2007)*, vol. 12, no. 20, 3194, 2007. DOI: https://doi.org/10.2807/esw.12.20.03194-en. [Online]. Available: https://www.eurosurveillance.org/content/10.2807/esw.12.20.03194-en.

[2] Centers for Disease Control and Prevention, *Antibiotic Resistance Threats in the United States, 2019*, 2019. DOI: 10.15620/cdc:82532. [Online]. Available: http://dx.doi.org/10.15620/cdc:82532..

[3] A. Cassini *et al.*, "Attributable deaths and disability-adjusted life-years caused by infections with antibiotic-resistant bacteria in the eu and the european economic area in 2015: A population-level modelling analysis," *The Lancet infectious diseases*, vol. 19, no. 1, pp. 56–66, 2019.

[4] C. J. Murray *et al.*, "Global burden of bacterial antimicrobial resistance in 2019: A systematic analysis," *The Lancet*, 2022.

[5] M. Renwick and E. Mossialos, "What are the economic barriers of antibiotic RD and how can we overcome them?" *https://doi.org/10.1080/17460441.2018.1515908*, vol. 13, no. 10, pp. 889–892, 2018, ISSN: 1746045X. DOI: 10.1080/17460441.2018.1515908. [Online]. Available: https://www.tandfonline.com/doi/abs/10.1080/17460441.2018.1515908.

[6] J. A. DiMasi, H. G. Grabowski, and R. W. Hansen, "Innovation in the pharmaceutical industry: New estimates of r&d costs," *Journal of health economics*, vol. 47, pp. 20–33, 2016.

[7] A. Paavola, *Big pharma backs off superbug: Why 5 drugmakers bailed on antibiotic research*, 2018. [Online]. Available: https://www.beckershospitalreview.com/pharmacy/big-pharma-backs-off-superbug-why-5-drugmakers-bailed-on-antibiotic-research.html.

[8] C. J. Clancy and M. H. Nguyen, "Buying time: The amr action fund and the state of antibiotic development in the united states 2020," in *Open Forum Infectious Diseases*, Oxford University Press US, vol. 7, 2020, ofaa464.

[9] K. Outterson *et al.*, "Accelerating global innovation to address antibacterial resistance: Introducing carb-x," *Nature Reviews Drug Discovery*, vol. 15, no. 9, pp. 589–590, 2016.

[10] P. J. Bergen *et al.*, "Synergistic killing of multidrug-resistant pseudomonas aeruginosa at multiple inocula by colistin combined with doripenem in an in vitro pharmacokinetic/pharmacodynamic model," *Antimicrobial agents and chemotherapy*, vol. 55, no. 12, pp. 5685–5695, 2011.

[11] C. Vidaillac, L. Benichou, and R. E. Duval, "In vitro synergy of colistin combinations against colistin-resistant acinetobacter baumannii, pseudomonas aeruginosa, and klebsiella pneumoniae isolates," *Antimicrobial agents and chemotherapy*, vol. 56, no. 9, pp. 4856–4861, 2012.

[12] M. Paul *et al.*, "Colistin alone versus colistin plus meropenem for treatment of severe infections caused by carbapenem-resistant gram-negative bacteria: An open-label, randomised controlled trial," *The Lancet Infectious Diseases*, vol. 18, no. 4, pp. 391–400, 2018.

[13] R. J. Melander and C. Melander, "The Challenge of Overcoming Antibiotic Resistance: An Adjuvant Approach?" *ACS Infectious Diseases*, vol. 3, no. 8, pp. 559–563, 2017, ISSN: 23738227. DOI: 10.1021/ACSINFECDIS.7B00071. [Online]. Available: https://pubs.acs.org/doi/abs/10.1021/acsinfecdis.7b00071.

[14] M. Pieren and M. Tigges, "Adjuvant strategies for potentiation of antibiotics to overcome antimicrobial resistance," *Current Opinion in Pharmacology*, vol. 12, no. 5, pp. 551–555, 2012, ISSN: 1471-4892. DOI: 10.1016/J.COPH.2012.07.005.

[15] S. M. Drawz and R. A. Bonomo, "Three decades of β-lactamase inhibitors," *Clinical microbiology reviews*, vol. 23, no. 1, pp. 160–201, 2010.

[16] D. Brown, "Antibiotic resistance breakers: can repurposed drugs fill the antibiotic discovery void?" *Nature Reviews Drug Discovery 2015 14:12*, vol. 14, no. 12, pp. 821–832, 2015, ISSN: 1474-1784. DOI: 10.1038/nrd4675. [Online]. Available: https://www.nature.com/articles/nrd4675.

[17] T. T. Talele, S. A. Khedkar, and A. C. Rigby, "Successful applications of computer aided drug discovery: Moving drugs from concept to the clinic," *Current topics in medicinal chemistry*, vol. 10, no. 1, pp. 127–141, 2010.

[18] J. M. Stokes *et al.*, "A Deep Learning Approach to Antibiotic Discovery Article A Deep Learning Approach to Antibiotic Discovery," *Cell*, vol. 180, 688–702.e13, 2020. DOI: 10.1016/j.cell.2020.01.021. [Online]. Available: https://doi.org/10.1016/j.cell.2020.01.021.

[19] L. David *et al.*, "Artificial intelligence and antibiotic discovery," *Antibiotics*, vol. 10, no. 11, p. 1376, 2021.

[20] W. Jin *et al.*, "Deep learning identifies synergistic drug combinations for treating covid-19," *Proceedings of the National Academy of Sciences*, vol. 118, no. 39, 2021.

[21] C. I. Bliss, "The toxicity of poisons applied jointly 1," *Annals of applied biology*, vol. 26, no. 3, pp. 585–615, 1939.

[22] K. Preuer, R. P. Lewis, S. Hochreiter, A. Bender, K. C. Bulusu, and G. Klambauer, "Deepsynergy: Predicting anti-cancer drug synergy with deep learning," *Bioinformatics*, vol. 34, no. 9, pp. 1538–1546, 2018.

[23] F. A. Bettelheim, W. H. Brown, M. K. Campbell, S. O. Farrell, and O. Torres, *Introduction to general, organic and biochemistry*. Cengage learning, 2012.

[24] M. Bertolini, L. Zhao, D.-A. Clevert, and F. Montanari, "Beyond atoms and bonds: Contextual explainability via molecular graphical depictions," 2022.

[25] A. Kulesa, J. Kehe, J. E. Hurtado, P. Tawde, and P. C. Blainey, "Combinatorial drug discovery in nanoliter droplets," *Proceedings of the National Academy of Sciences*, vol. 115, no. 26, pp. 6685–6690, 2018.

[26] C. Bliss, "The calculation of microbial assays," *Bacteriological reviews*, vol. 20, no. 4, pp. 243–258, 1956.

[27] Z. Wu *et al.*, "Moleculenet: A benchmark for molecular machine learning," *Chemical science*, vol. 9, no. 2, pp. 513–530, 2018.

[28] S. M. Corsello *et al.*, "The drug repurposing hub: A next-generation drug library and information resource," *Nature medicine*, vol. 23, no. 4, pp. 405–408, 2017.

[29] A. Vaswani *et al.*, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[30] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," *arXiv preprint arXiv:1710.10903*, 2017.

[31] A. Deac, Y.-H. Huang, P. Veličković, P. Liò, and J. Tang, "Drug-drug adverse effect prediction with graph co-attention," *arXiv preprint arXiv:1905.00534*, 2019.

[32] J. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl, "Algorithms for hyper-parameter optimization," *Advances in neural information processing systems*, vol. 24, 2011.

[33] A. L. Maas, A. Y. Hannun, A. Y. Ng, *et al.*, "Rectifier nonlinearities improve neural network acoustic models," in *Proc. icml*, Atlanta, Georgia, USA, vol. 30, 2013, p. 3.

[34] A. Lydia and S. Francis, "Adagrad—an optimizer for stochastic gradient descent," *Int. J. Inf. Comput. Sci*, vol. 6, no. 5, pp. 566–568, 2019.

[35] M. D. Zeiler, "Adadelta: An adaptive learning rate method," *arXiv preprint arXiv:1212.5701*, 2012.

[36] J. Ross, B. Belgodere, V. Chenthamarakshan, I. Padhi, Y. Mroueh, and P. Das, "Large-scale chemical language representations capture molecular structure and properties," *Nature Machine Intelligence*, vol. 4, no. 12, pp. 1256–1264, 2022.

[37] Z. Xiong *et al.*, "Pushing the boundaries of molecular representation for drug discovery with the graph attention mechanism," *Journal of medicinal chemistry*, vol. 63, no. 16, pp. 8749–8760, 2019.

[38] X. Wang, Z. Li, M. Jiang, S. Wang, S. Zhang, and Z. Wei, "Molecule property prediction based on spatial graph embedding," *Journal of chemical information and modeling*, vol. 59, no. 9, pp. 3817–3828, 2019.

[39] K. Yang *et al.*, "Analyzing learned molecular representations for property prediction," *Journal of chemical information and modeling*, vol. 59, no. 8, pp. 3370–3388, 2019.

[40] H. Altae-Tran, B. Ramsundar, A. S. Pappu, and V. Pande, "Low data drug discovery with one-shot learning," *ACS central science*, vol. 3, no. 4, pp. 283–293, 2017.

[41] *Mometasone furoate*. [Online]. Available: https://go.drugbank.com/drugs/DB14512.

[42] A. Neher, M. Gstöttner, A. Scholtz, and M. Nagl, "Antibacterial activity of mometasone furoate," *Archives of Otolaryngology–Head & Neck Surgery*, vol. 134, no. 5, pp. 519–521, 2008.

[43] M. Pitcairn, J. Schuler, P. R. Erve, S. Holtzman, and W. Schumer, "Glucocorticoid and antibiotic effect on experimental gram-negative bacteremic shock," *Archives of Surgery*, vol. 110, no. 8, pp. 1012–1015, 1975.

[44] S. Asano, M. Komiya, N. Koike, E. Koga, S. Nakatani, and Y. Isobe, "5, 6, 7, 8-tetrahydropyrido [4, 3-d] pyrimidines as novel class of potent and highly selective camkii inhibitors," *Bioorganic & medicinal chemistry letters*, vol. 20, no. 22, pp. 6696–6698, 2010.

[45] F. Chi *et al.*, "Vimentin-mediated signalling is required for ibea+ e. coli k1 invasion of human brain microvascular endothelial cells," *Biochemical Journal*, vol. 427, no. 1, pp. 79–90, 2010.

[46] S. Giunta, L. Galeazzi, G. Turchetti, G. Sampaoli, and G Groppa, "In vitro antistreptococcal activity of the potassium-sparing diuretics amiloride and triamterene," *Antimicrobial agents and chemotherapy*, vol. 28, no. 3, pp. 419–420, 1985.

[47] S. Sugiyama, E. J. Cragoe Jr, and Y Imae, "Amiloride, a specific inhibitor for the na+-driven flagellar motors of alkalophilic bacillus.," *Journal of Biological Chemistry*, vol. 263, no. 17, pp. 8215–8219, 1988.

[48] M. Islam *et al.*, "Novel amiloride derivatives that inhibit bacterial motility across multiple strains and stator types," *Journal of Bacteriology*, vol. 203, no. 22, e00367–21, 2021.

[49] J. Doukas *et al.*, "Phosphoinositide 3-kinase γ/δ inhibition limits infarct size after myocardial ischemia/reperfusion injury," *Proceedings of the National Academy of Sciences*, vol. 103, no. 52, pp. 19 866–19 871, 2006.

[50] Y. Yang, K. Zha, Y. Chen, H. Wang, and D. Katabi, "Delving into deep imbalanced regression," in *International Conference on Machine Learning*, PMLR, 2021, pp. 11 842–11 851.

[51] D. Gurvic, A. G. Leach, and U. Zachariae, "Data-driven derivation of molecular substructures that enhance drug activity in gram-negative bacteria," *Journal of Medicinal Chemistry*, vol. 65, no. 8, pp. 6088–6099, 2022.