University of Southampton

Faculty of Physical Sciences and Engineering

Electronics and Computer Science

Graph Convolution on Molecular Graphs For The Classification of Toxic Molecules

by

Mortimer Sotom

Supervisor: Pr. Mahesan Niranjan
Second Examiner: Pr. Mark S. Nixon

A dissertation submitted in partial fulfilment of the degree of

MSc Artificial Intelligence

Statement of Originality

- I have read and understood the ECS Academic Integrity information and the University's Academic Integrity Guidance for Students.
- I am aware that failure to act in accordance with the Regulations Governing Academic Integrity may lead to the imposition of penalties which, for the most serious cases, may include termination of programme.
- I consent to the University copying and distributing any or all of my work in any form and using third parties (who may be based outside the EU/EEA) to verify whether my work contains plagiarised material, and for quality assurance purposes.

I have acknowledged all sources, and identified any content taken from elsewhere.

I have not used any resources produced by anyone else.

I did all the work myself, or with my allocated group, and have not helped anyone else.

The material in the report is genuine, and I have included all my data/code/designs.

I have not submitted any part of this work for another assessment.

My work did not involve human participants, their cells or data, or animals.

Abstract

The drug discovery process requires chemists to synthesize in labs, molecules which they believe inhibit desirable properties. This costly and time consuming process can benefit from machine learning models which can predict molecular properties of molecules. Recent advancements made to Convolutional Neural Network have extended their applications to graphs leading to the design of Graph Convolutional Networks (GCNs). An analysis of the literature is presented to explain the underlying theory behind graph convolution. The convolution, pooling and translation operators are only defined for regular grids so these must be redefined in the spectral domain. However, graph coarsening does not make sense when applied to molecules so alternate methods are considered such as the creation of dummy super-nodes. From the techniques identified, a GCN is developed and trained on the Tox21 dataset to classify molecules as potentially toxic. Additionally, machine learning models remain mostly black boxes which reduces the adoption rate of such models in applications involving human health and safety. A discussion highlights the importance of explaining the rationale behind a model's predictions for users to establish a level of trust and confidence in the model. It is demonstrated how such explanations can be obtained with the LIME software interpreting a Neural Network which encodes molecules as sub-structures (via Extended-Connectivity Fingerprints). This experimental set up was able to identify which molecular fragments the model believes are responsible for the toxicity of molecules. Finally, this piece identifies a gap in the literature and explains how this project can be extended to address it.

Acknowledgements

I would like to present my special thanks for the guidance of my supervisor Pr. Niranjan Mahesaon for the entire duration of this dissertation.
I am also thankful to the University of Southampton in supporting this project by providing access to the Iridis 5 supercomputer.

Table of Contents

List of Figures

List of Tables

Abbreviations

ADMET: Absoption, distribution, metabolism, excretion and toxicity

AUC: Area Under Curve

ECFP: Extended-Connectivity Fingerprints CNN: Convolutional Neural Network

GCN: Graph Convolutional Neural Network

LIME: Local Interpretable Model-Agnostic Explanations software

NN: Neural Network

ROC: Receiver Operating Characteristic

SMILES: Simplified Molecular-Input Line-Entry System

Chapter 1

Introduction

1.1 Introduction

The rapid growth of available data and the computational power of graphics processing units (GPUs) have led machine learning models to achieve high performances on a range of problems in multiple domains. In the last few years, extensive studies have been conducted on convolutional neural networks (CNNs) due to their powerful ability to solve high dimensional learning problems. CNNs can leverage the compositionality structure of data to extract compositional features and feed them to classifiers or regressors, achieving state-of-the-art results in a variety of tasks, including natural language processing, speech recognition and visual recognition [1]. CNNs are considered to be the *swiss knife of computer science* due to their wide range of applications, and research is now focusing on extending this to be applicable on graphs.

In parallel to these advancements, the domain of chemistry quickly evolved by incorporating machine learning techniques which gave rise to the field of chemoinformatics. With the development of machine learning models came computational chemistry, which attempted to predict chemical phenomena through deductive learning [2]. Johann Gasteiger who is considered to be a pioneer in chemoinformatics defines the field simply as "the application of informatics methods to solve chemical problems" [3]. Neural Networks (NNs) have been successfully implemented to predict the molecular properties of compounds which has been a huge aid for the drug design and drug discovery processes. However, the state-of-the-art encoding method previous to graph convolution, Extended-Connectivity Fingerprints (ECFP), suffers from information loss for reasons explained in greater detail in section 2.1.1, which in turn limits the predictive performance, accuracy and reliability of machine learning models.

The five properties of drug absorption, distribution, metabolism, excretion and toxicity (ADMET), are the main performance influences on the pharmacological activity of drugs. Tight regulations are imposed on these properties for potential drugs so chemists must synthesize in labs molecules, which they believe inhibit the right levels of these properties from empirical deduction, to confirm their predictions. However this is a highly time-consuming and expensive process which delays drug discovery. Over 30% of pharmaceuticals fail human clinical trials due to toxicity levels which weren't identified during pre-clinical studies in animal models [4]. Hence the interest of applying machine learning models to such a task [5].

In order for NN based models to be brought to applications which are related to human health and safety, a major weakness of NNs must be overcome: explain their decision making process. The rationale behind a model's predictions need to be understood for the results to be credible. Measuring the performance of the model with a metric such as accuracy is not sufficient as that metric is limited to the training and testing sets used. Users must be confident in the model and trust the results before letting their decisions be influenced. This is especially important in sectors such as medical diagnosis, chemical research and even terrorism detection as decisions can have critical impacts.

In many real life problems, data cannot be represented as regularly shaped tensors but are better structured as graphs. Graphs are data representations which do not lie in Euclidean domains such as point clouds, social networks and chemical molecules. However, the compositionality and stationarity properties which allow kernel-based convolutions to operate on regular grids are not defined on graphs. Hence, the convolution and pooling operators must be redefined to generalize CNNs to graphs to construct Graph Convolutional Networks (GCNs). This presents multiple challenges theoretically and implementation wise which are discussed in chapter 3.

1.2 Aims & Objectives

Extending convolutions to graphs is an emerging but promising field of research. The primary aim of this project is to understand and explain the underlying theory behind graph convolution. Once the fundamentals of GCNs is understood, the secondary aim is to apply a GCN on a chemical dataset and predict certain properties of molecules as a classification task.

The overall project can be broken down into four main objectives:

- Present an analysis of the various techniques and approaches described in the literature for graph convolution.
- Develop a Graph Convolutional Network.
- Interpret and explain the outputs of the model.
- Identify which molecular features or sub-structures are activated during the classification of molecules.

Chapter 2

Background

The application of graph convolution to molecules is an inter-disciplinary field of research so some background knowledge is required before diving into the specifics of GCNs which builds on these fundamentals. This chapter aims to provide all the necessary information to understand the concepts presented. The areas covered are the chemistry notations for molecules, the previous state-of-the-art method for encoding molecules and how this information is encoded for graphs. Finally, the basic architecture and operations of CNNs are reviewed.

2.1 Molecular Representation

There is a strong correlation between the structure of molecules and their physical, chemical and biological properties [6]. It is therefore critical for molecules to be represented and encoded in a manner which maintains these correlations, and in a format interpretable by machine learning models. The latter sets the main challenge in chemoinformatics as learning models expect homogeneous inputs but molecular graphs are heterogeneous among molecules.

Chemical molecules and reactions are generally represented by a universal line notation called SMILES (Simplified Molecular-Input Line-Entry System) developed in the late 1980s [7]. It is not a computer data structure but a linguistic construct detailing the atoms present in the molecule and their bond types in sufficient detail to generate its molecular structure.

Figure 2.1 illustrates the 2D and 3D molecular graphs of Vanillin and Nicotine generated from the SMILES notation. Usually the same molecule can be represented by multiple valid generic SMILES depending on the first and last atoms chosen for the line notation and the complexity of the molecule. A unique SMILES can be generated from all valid notations using a canonicalisation algorithm [8].

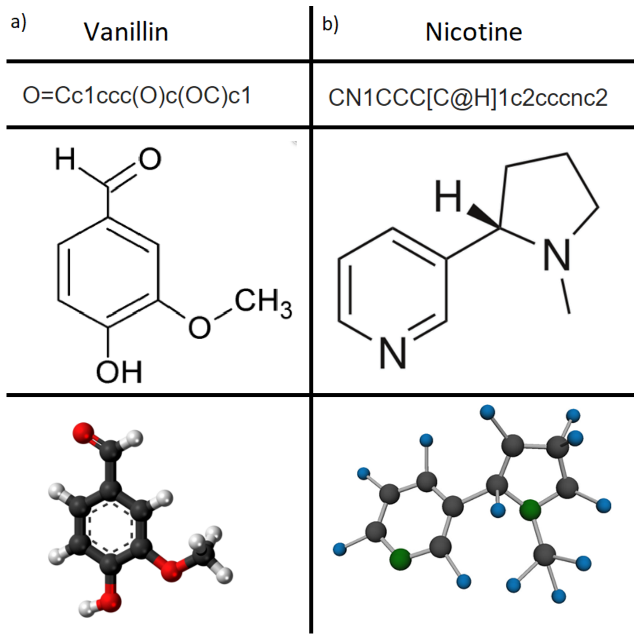


Figure 2.1: The SMILES notations of Vanillin (a) and Nicotine (b) compounds with their respective 2D and 3D molecular graphs.

A step referred to as encoding, translates this symbolic notation into fixed length vectors or matrices, depending on the technique employed, which hold the relevant information about the molecule. ECFP is a technique which encodes all molecular information as vectors of variable lengths while graph-convolution based models require inputs in the form of matrices. Automated open-source softwares such as RDKit in Python [9] are available to perform these types of encoding with some level of customization depending on the application.

2.1.1 Extended-Connectivity Fingerprints

ECFPs [10] are one of the most popular methods to encode molecules. Molecular data is encoded into fixed length vectors called fingerprints rendering them ideal for input into predictive models and require little to no configuration [11]. ECFPs encode the environment of atoms in a compound to a key-based or hashed identifier. Each bit or identifier in a fingerprint represents the absence or presence of a certain molecular feature. There are multiple types of fingerprints which store different information but the first and most important is the structural one. Each bit of the fingerprint is dedicated to a specific molecular sub-structure, also known as a fragment. As illustrated in Figure 2.2, if that fragment is present in the molecule being encoded, then the according bit is set to 1 and set to 0 otherwise. The remaining fingerprints describe the chemical information of those sub-structures activated in the structural fingerprint. These are automatically generated as all sub-structures of the structural fingerprint are already pre-defined.

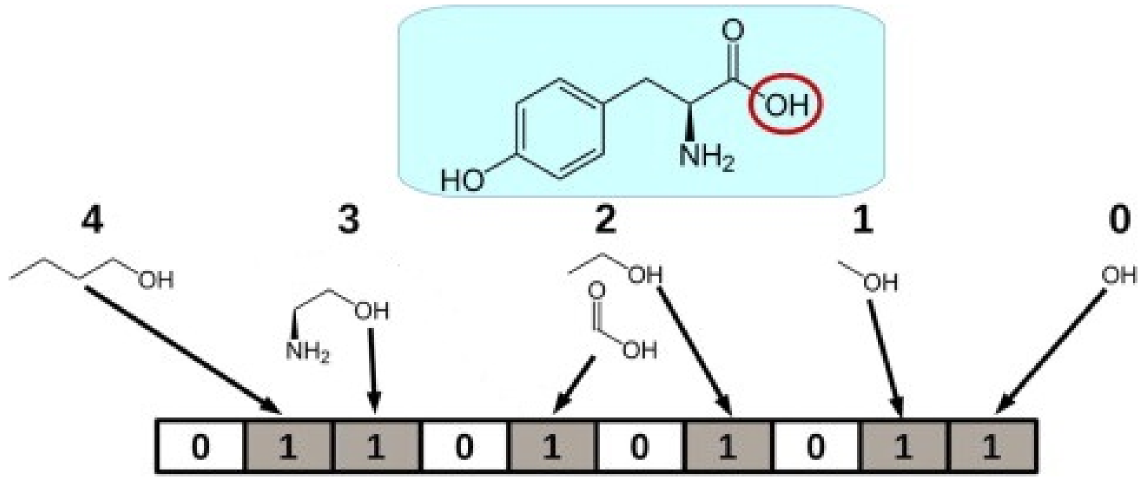


Figure 2.2: A hypothetical 10 bit structural fingerprint encoding the structure of an example molecule. The circled fragment is the starting point for the encoding process and the indices above the sub-structures represent their distance to the original fragment. The bits associated with sub-structures present in the molecule are then set to 1 and the remaining ones are set to 0. [11].

ECFPs have become a standard for encoding molecules in chemoinformatics. RDKit provides this functionality; the fingerprints can be obtained simply by providing the SMILES notation of the molecule to be encoded. Optional parameters allow for some degree of customization such as specifying the length of the vectors. Figure 2.3 highlights the ease of use of this technique with a classic feedforward network. The user inputs the SMILES notation of the molecules in the dataset to obtain the ECFPs with RDKit, which in turn are inputted into a classic NN for a regression or classification task.

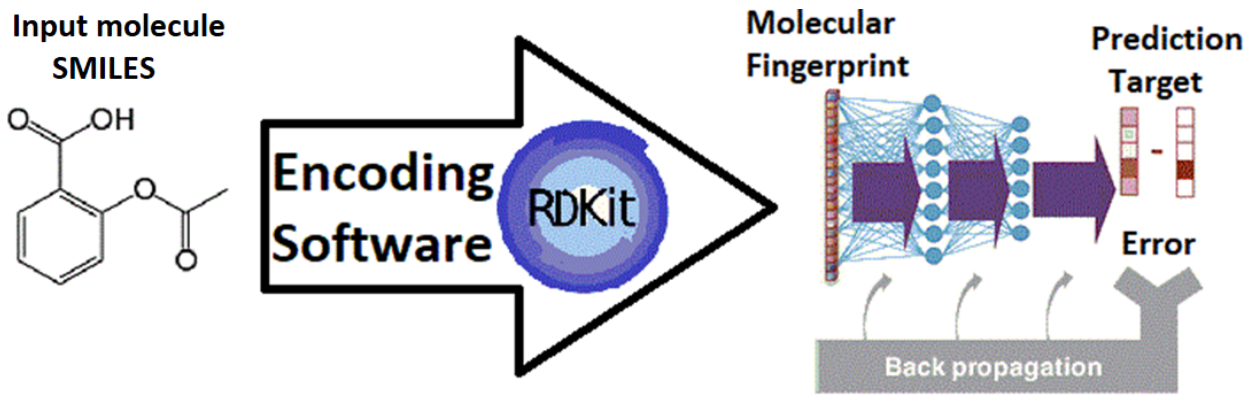


Figure 2.3: Workflow of using ECFPs with NNs. Molecules are inputted as SMILES, ECFPs are obtained with RDKit, ECFPs are inputs to a feedforward NN for regression or classification tasks. Image modified from: [12]

The encoding method of ECFPs unfortunately suffers from two fundamental issues. The first being that structural fingerprints tend to be very large in an attempt to capture as many sub-structures of the molecule as possible. This results in the vector being very sparse leading to memory issues with some models and requiring higher computational power [5]. The second issue is that the other fingerprints hold information about the fragments of the structural fingerprint. This signifies that only information of small neighbourhoods are encoded as opposed to the molecule as a whole. This entails that there is some information loss about the molecule which will limit the accuracy and reliability of the model used. Furthermore, the discovery of features is then restricted to compositions of the fragments specified in the structural fingerprint [13]. A solution to these problems is to extract all the relevant information directly from the molecular graph, via graph convolution.

2.1.2 Graph Encoding for Convolution

A graph is described by its topological structure and feature vectors held at each node as show in Figure 2.4. A graph can be defined mathematically by $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is the set of vertices with $|\mathcal{V}| = N$ nodes and $\mathcal{E}$ is the set of edges. Graphs can be directed or undirected with the only difference being edges have a direction in directed graphs, they are unilateral, while edges in undirected graphs can be interpreted as being bilateral. Since molecules are undirected graphs, directed graphs will not be considered.

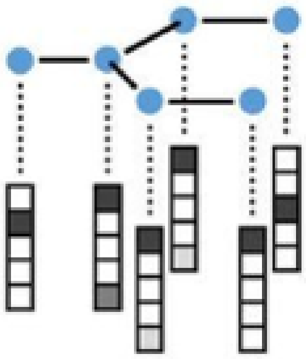


Figure 2.4: A graph with 5-bit feature vectors at each node.

The following explanation can be followed with Figure 2.5. The topological structure of a graph is represented by an adjacency matrix $A = [a_{ij}] \in \mathbb{R}^{N \times N}$ which encodes connections between nodes. All nodes are indexed ascendingly such that $[a_{ij}]$=1 denotes an edge between nodes i and j. Alternatively, a graph can also be represented by an adjacency list where entries at index i list the index of nodes connected to node i. The same results can be obtained with either adjacency matrices or adjacency lists as they represent the same information [14]. There is no standard for which notation to use but adjacency matrices are more widely due to a simpler index layout. Finally, let $D = diag(d_1, d_2, ..., d_N)$ be the diagonal matrix of A where $d_i = \sum_j a_{ij}$ is the degree, number of connections, of node i.

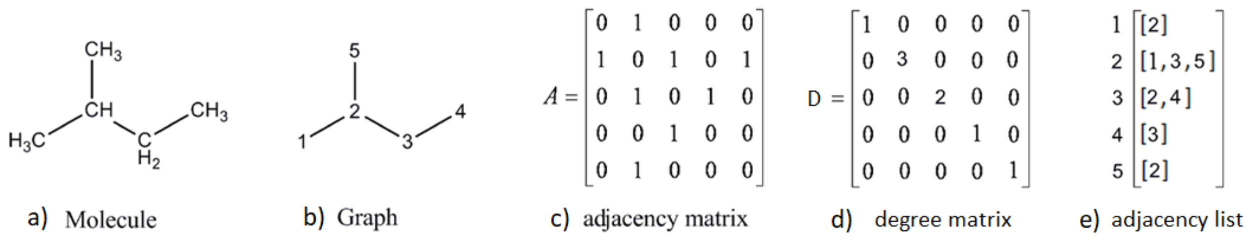


Figure 2.5: a) Example molecular graph without Hydrogen bonds. b) Graph with numbered nodes. c) Adjacency matrix representation of the graph. d) Degree matrix obtained from the adjacency matrix. e) Adjacency list representation of the graph.

The information of any graph is held at node-level by feature vectors. These feature vectors share the same labels between all nodes and are of identical length. These features are usually one-hot encoded in preparation to use as input into machine learning models. For example, a feature vectors for molecules can hot-hot encode the features of each atom with element type, electric charge, number of valence elctrons, etc . All feature vectors can then be concatenated vertically to create a graph feature matrix. In this manner, all information about the graph is summarised as an adjacency matrix and a feature matrix.

2.2 CNNs Basics

In order to extend convolution from images to graphs, the basic operations of CNNs must first be well understood. Figure 2.6 shows a standard CNN architecture with a 32×32 pixel image. Once an initial set of small filters are chosen, the convolution operation follows as the dot product of the filter with the receptive field, to obtain a one pixel output of a feature map. The translation operator determines the 'strides', the direction in which the convolution filter moves across the image in order to output all pixels of the feature map. On a regular grid this is simply moving the filter by one pixel across the width of the image and then repeating this step one pixel down, until the entire image is covered. The pooling operation aims to reduce the dimensionality of the feature maps while retaining the most important information. A pooling filter is applied on the feature maps and outputs the mean, median or max value of the receptive field. The translation operator still defines how the pooling filter moves across the image but the stride length is equal to the filter size.

The power of CNNs comes from their ability to leverage the compositionality structure of data to extract its compositional features. The convolution operation preserves the spatial relationship between pixels by using filters of size greater than 1×1, making the convolution filters shift-invariant. With each convolution and pooling operations, the deeper the network becomes, the filters are able to learn higher abstract levels of representation until the dimensions of the data is reduce to that of a vector. This vector can then be used as input into a classifier.

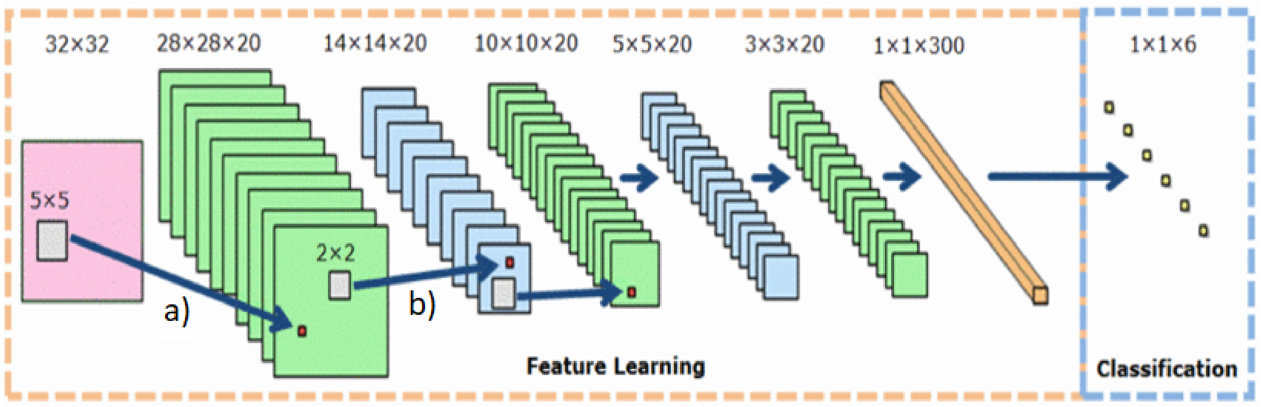


Figure 2.6: Example CNN architecture for a 32x32 pixel image with a 5x5 filter for the first convolution operation (a) and a 2x2 filter for the first pooling operation (b) for a 6 task classification. [15]

Chapter 3

Literature Review

The generalisation of CNNs to graphs is not evident as the convolution, pooling and translation operators are only defined for regular grids. This presents challenges theoretically and implementation-wise. Extending convolution to graphs has been discussed for many years with research focusing on label propagation [16], spectral graph clustering assumptions [17] and node feature embedding [18] [19], but only in the last few years have theories developed enough to obtain results comparable or superior to previous state-of-the-art methods. However there is still some confusion within the community regarding the proper working mechanisms for graph convolution [20]. Many papers claim to implement 'graph convolution' but actually tackle the task differently which can follow one of two strategies: a spectral approach or a spatial approach. Both start with the representation of a graph as a topological structure and each node having a feature vector of fixed length. The next section covers how the convolution operation is defined in both approaches and briefly describes the pooling operation in the spatial approach. The pooling operation in the context of the spectral approach is presented in a separate section as the domain actually reverts to a spatial representation. The overall graph convolution process explained in this chapter is summarised in Figure 3.1.

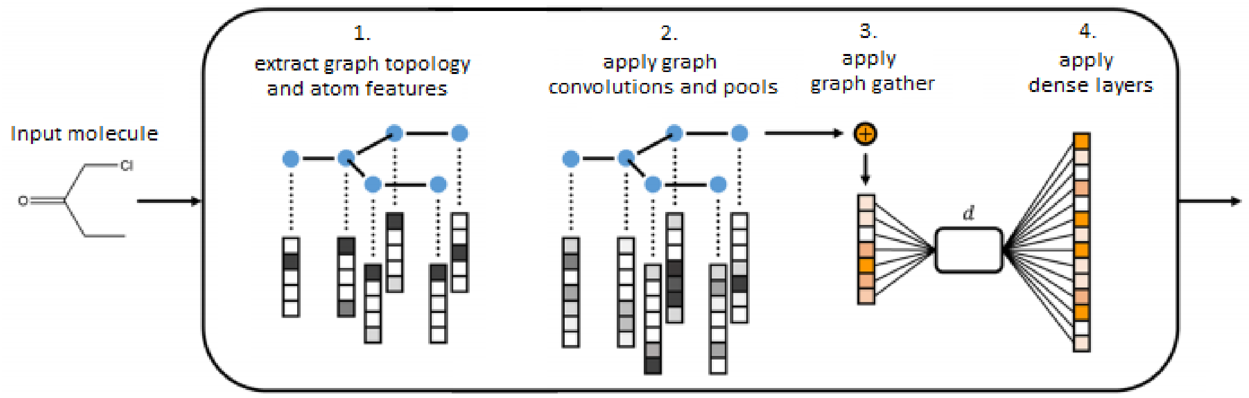

Input molecule
Cl
O
1. extract graph topology and atom features
2. apply graph convolutions and pools
3. apply graph gather
4. apply dense layers
d

Figure 3.1: Overview of GCNs from input to output. [21]

3.1 Convolution Operation

3.1.1 Spatial Approach

A spatial approach seems more intuitive at first since it can be compared to CNNs, with the main difference being that the data points (pixels on images and nodes on graphs) aren't connected in a grid like fashion. By construction, spatial approaches aim to reproduce filter localisation by defining a fixed sized kernel. The kernel would construct a new feature vector for each node in the graph by convolving its neighbouring feature vectors. Although conceivable, it must overcome the challenges of matching local neighbourhoods to the kernel, sub-graph matching (see Figure 3.2), and defining a translation operator to apply the kernel across the graph. The sub-graph matching problem is outlined in [19] and the difficulty of identifying identical localised patterns across different graphs is discussed in [22] and [23]. The translation operator must be defined such that the kernel is applied across the graph without missing a node or convolving the same node twice [22]. Since the topology of graphs is extremely variable, there is no unique mathematical definition of translation on graphs from a spatial perspective, so this is achieved with algorithms explained in [24].

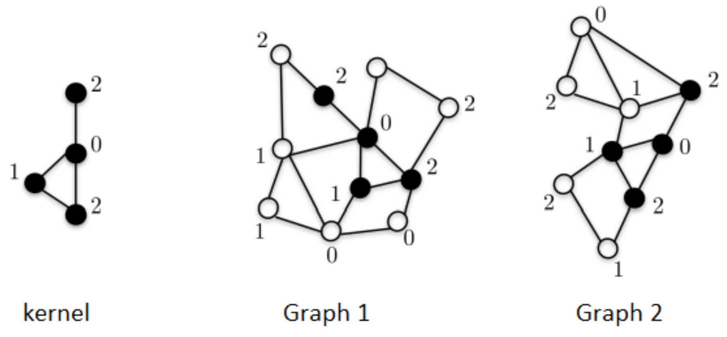


Figure 3.2: Sub-graph matching problem. [24]

The next step in spatial graph convolution is mimicking the pooling operation in CNNs which reduces the size of the grid. When applied to graphs, the objective of the pooling operation is to gather nodes with similar local features to cluster them together and form a new graph where the information is more dense. This effectively shapes the challenge of designing a multi-scale coarsening algorithm that preserves local geometric structures of the graph [22] [23]. An example of a multiresolution clustering of a graph is provided in appendix A. However, graph clustering is NP-hard [25] so approximations are required by using balance cuts, powerful combinatorial graph partitioning models [26] and heavy-edge-matching.

In the context of applying a GCN on a molecule based dataset, graph coarsening is illogical as altering the topological structure of the graph would completely change the molecule and its features. For this reason and due to the complexity of this approach, it was not considered further for the implementation of this project.

3.1.2 Spectral Approach

The work of Bruna et a. (2014) [22] formulated the basis of spectral networks by drawing from the convolution properties in the Fourier domain. The generalisation of convolution to graphs was derived from its definition given by the convolution theorem [27], as linear operators diagonalised by the Fourier basis, which can be identified by the eigenvectors of the graph Laplacian operator. From graph theory, the combinatorial graph Laplacian matrix $L = D - A$, where A is the adjacency matrix of the graph and D its degree matrix as explained in chapter 2.1.2. The normalised version of the graph Laplacian is the generalised form of the convolution operator to graphs:

$$\widetilde{L} = I - D^{-\frac{1}{2}}\, A\, D^{-\frac{1}{2}} \tag{3.1}$$

The node feature vectors attributed to a graph G with N nodes, are decomposed as signals $x \in \mathbb{R}^N$ and convolved with a spectral filter $g_\theta = diag(\theta)$. U represents the matrix of eigenvectors of L and $\theta \in \mathbb{R}^N$ is the vector of Fourier coefficients such that the filters g_θ are multipliers of the eigenvalues of L. ie:

$$g \star x = U\, g_\theta\, U^T\, x \tag{3.2}$$

However, two issues arise with a filter define in the spectral domain: (i) it is not localised in space and (ii) the calculation of equation 3.2 is computationally expensive in the order of $O(N^2)$ due to the multiplication of matrix of eigenvectors U. To overcome these limitations, Defferrard et al (2016) [23] approximate the spectral filter g_θ with the Chebyshev polynomials recursive series $T_k(x)$ defined on the graph Laplacian up to K^{th} order as suggested in [28]. The equation now takes the following form where θ_k is the vector of Chebyshev coefficients (weights):

$$g \star x = \sum_{k=0}^{K} \theta_k\, T_k(\widetilde{L})\, x \tag{3.3}$$

The model was finally further simplified by Kipf et al. (2017)[29] by limiting K=1 and approximating the largest eigenvalue of L by 2 for a layer-wise linear function with regards to L. This reduces overfitting on graphs with a wide range of node degrees. The last convolution equation in the spectral domain is:

$$g \star x = \theta\, (I + D^{-\frac{1}{2}}\, A\, D^{-\frac{1}{2}})\, x \tag{3.4}$$

On a final note, $I + D^{-\frac{1}{2}} A D^{-\frac{1}{2}}$ has eigenvalues with a range of [0-2], hence, frequent use of this operator leads to the exploding/vanishing gradient problem. This is solved by [29]'s *renormalization trick* of the Laplacian allowing for deep networks to be constructed:

$$\widehat{L} = \widetilde{D}^{-\frac{1}{2}}\,\widetilde{A}\,\widetilde{D}^{-\frac{1}{2}} \tag{3.5}$$

Where $\widetilde{A} = A + I$ and $\widetilde{D}_{ii} = \sum_j \widetilde{A}_{ij}$

Convolution in the spectral domain is performed element wise so $g \star x$ must be repeated for each element in the feature vectors. At this point, equation 3.4 can be formulated in a spectral free expression for the convolution operation to be generalised in the spatial domain by equation 3.6 for graphs with an adjacency matrix $A \in \mathbb{R}^{N \times N}$ and a feature matrix $X \in \mathbb{R}^{N \times F}$. $\theta \in \mathbb{R}^{F \times Out}$ is a matrix of trainable parameters (weights) such that the output convolved matrix $Z \in \mathbb{R}^{N \times Out}$.

$$Z = \widetilde{D}^{-\frac{1}{2}}\,\widetilde{A}\,\widetilde{D}^{-\frac{1}{2}}\,X\,\theta \;= \widehat{L}\,X\,\theta \tag{3.6}$$

This convolution operation in equation 3.6 can be visualised for a single node in Figure 3.3 to better understand the individual steps involved. Consider a node v with three neighbouring nodes which are identified by the adjacency matrix A. The feature vectors of all four nodes are multiplied by their respective weights $(X\theta)$. They feature vectors of all neighbouring nodes are summed to create the new feature vector of node v which is matrix multiplication of $\widehat{L}\,X$. Finally the vector is updated by the activation function of the GCN model.

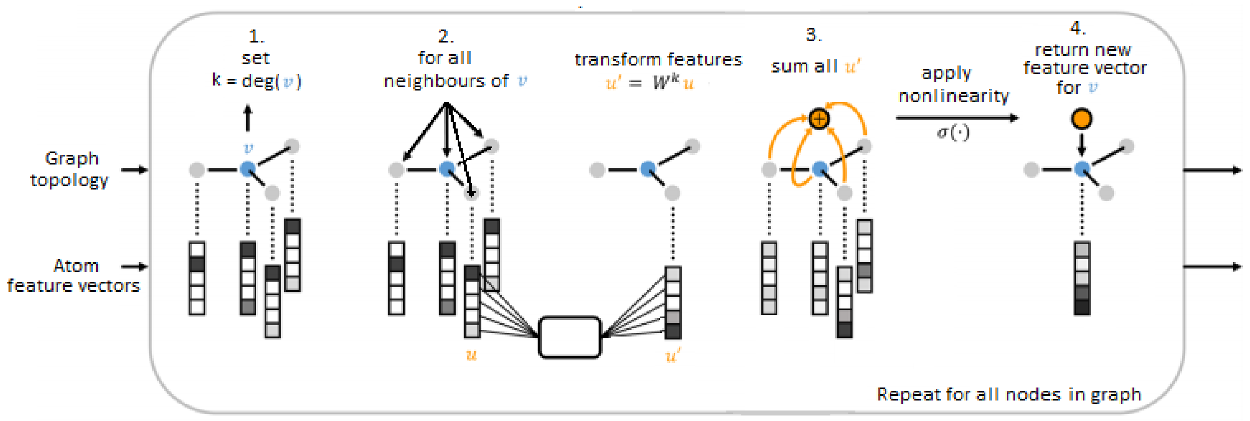

Graph topology
Atom feature vectors
1. set k = deg(v)
v
2. for all neighbours of v
u
transform features $u' = W^k u$
u'
3. sum all u'
apply nonlinearity
$\sigma(\cdot)$
4. return new feature vector for v
Repeat for all nodes in graph

Figure 3.3: Visualisation of the convolution steps for one node. Image modified from: [21]

3.2 CNNs to GCNs

Now that the convolution operation has been redefined on graphs, the same must be done for the pooling operation in a manner which does not alter the topological structure of the gaph. It must also be noted that these convolution and pooling operations manipulate the node feature vectors but an overall representation of the graph is required to obtain graph-level outputs for classification and regression tasks. Yet again, the literature detailing these operations is poor, only providing high level explanations which are not sufficient for a proper implementation such as "a global pooling step combines features from all the atoms in the molecule" [30]. The following explanations were compiled by scrutinizing codes of similar works, [31] [32] [33], made available on Github and by contacting the author of one of these projects [29], Thomas N. Kipf.

When comparing pooling on images to graphs, the neighbouring nodes of a single node v can be interpreted as the receptive field for the pooling filter. The pooling operation is performed feature-wise. In the case of max pooling, feature x_i of node v, is replaced by the maximum of value of the i-th feature of the neighbouring nodes and itself. This is repeated for all features in the feature vector and for every node. The graph gather operation aims to transform the information of graph represented from node-level to graph-level by summing all the feature vectors of the graph together. Once the graph can be represented by a single vector, it can be inputted into a classifier or regressor. The graph pooling and graph gather operations are illustrated in Figure 3.4.

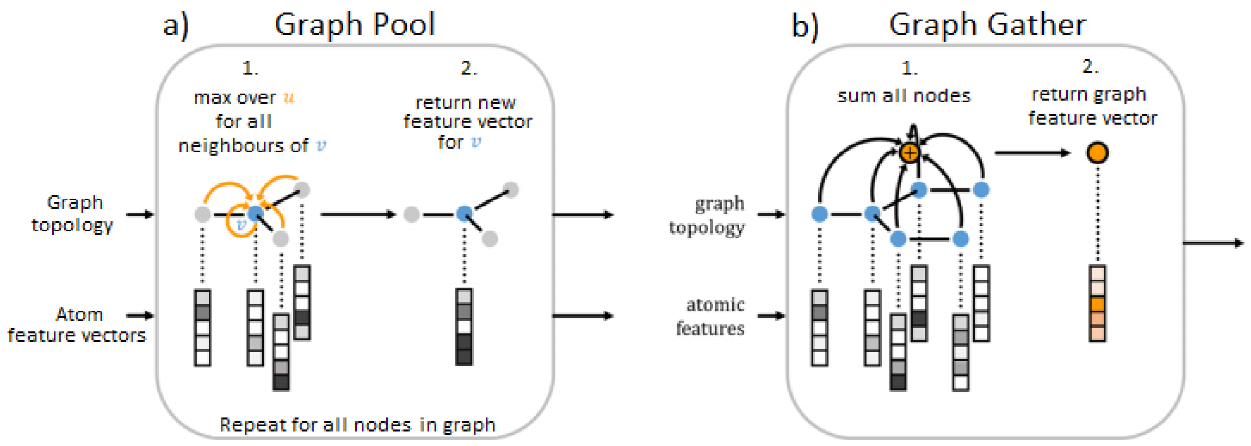

a) Graph Pool
1. max over u for all neighbours of v
2. return new feature vector for v
Graph topology
Atom feature vectors
Repeat for all nodes in graph
b) Graph Gather
1. sum all nodes
2. return graph feature vector
graph topology
atomic features

Figure 3.4: a)Graph pooling for one node. b) Graph Gather for graph-level representation as a single vector. Image modified from: [21]

Scarselli et al. (2011) [19] suggested the creation of a dummy 'super-node', Sn, which is connected to all nodes in the graph such that $deg(Sn) = N$ as shows in Figure 3.5. This allows for graph-level classification/regression to be managed in the same fashion as node-level classification/regression by implicitly merging the graph-gather step into the convolution operation [29]. The feature vector of the super-node is initialised with 0s and the adjacency matrix of the graph must be updated to have (N+1) dimensions to include the super-node. There is some debate over this technique as it does not include a pooling operation and therefore a higher level, more dense representation of the graph can not be obtained. This is especially relevant when graph coarsening is implemented but would have a minimal impact on the classification results for graphs which don't have their topology altered such as with molecules [33]. These papers briefly discuss the differences of these techniques but no research has been conducted to investigate how they affect the performance of GCN models! This is clear gap in the literature which is proposed as an extension to this project for future work in chapter 5.3.

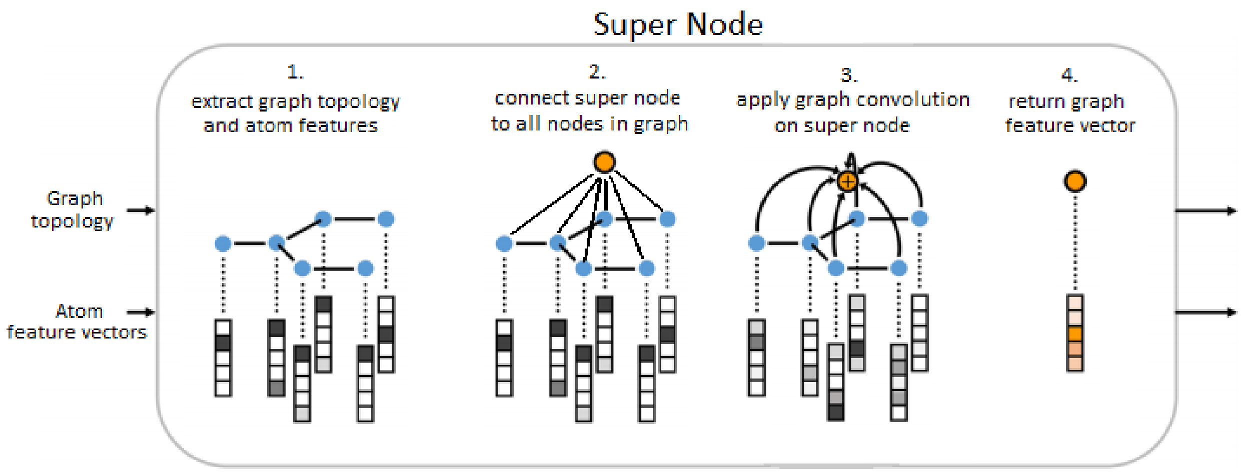

Super Node
1. extract graph topology and atom features
2. connect super node to all nodes in graph
3. apply graph convolution on super node
4. return graph feature vector
Graph topology
Atom feature vectors

Figure 3.5: Creation of super-node which combines graph convolution and graph gather into a single operation. Image modified from: [21]

3.3 Other Approaches

Besides from the works referenced in the previous section, there are some honourable mentions which study the problem from a different perspective but achieve comparable results. To remain concise, only projects working with molecules are mentioned and are not explained in great detail.

S. Kearnes et al. (2016) [32] decided to have two different graph feature matrices. The adjacency matrix is still used to represent the structure of the molecule and the standard feature matrix is also included and referred to as the Atom-layer. The second feature matrix consistutes the Pair-layer which encodes atom pairs to capture extra information such as the type of bonds connecting the atoms and the strength of that bond. The convolution operation is applied between Atom-layers, between Pair-layers and between both layers to construct a Weave module.

MoleculeNet [31] is an exceptional project with the intention of creating a benchmark for molecular machine learning. It implements 16 datasets covering the four main branches of chemistry with 6 featurisation methods which includes ECFP, Weave modules and graph convolution. All methods and datasets are integrated as parts of the open-source DeepChem package for Python [14].

Chapter 4

Methodology

This chapter describes the methodology and experimental setup used to obtain the results discussed in chapter 5. The first section presents the dataset used along with how the chemical information constituting the feature vectors was obtained as it is not provided. The architecture of the model is then explained with the chosen hyperparameters and how the technique implemented allows for graph-level classification. The chapter concludes by explaining how softwares such as LIME are able to extract explanations from a model, clarifying how the model makes certain predictions.

4.1 Datasets

The "Toxicology in the 21st Century" (Tox21) [4] program founded in 2008, assembled a public database of toxic compounds to promote research of methods to assess chemical toxicity. The database is updated frequently and now contains qualitative toxicity measurements for over 10,000 chemicals and drugs on 12 targets related to stress response pathways and nuclear receptors. This dataset was also used for the 2014 Tox21 Data Challenge with the goal of crowd-sourcing data analysis by independent researchers [34]. It has since been used as a benchmark to assess the performance of new models since many research papers use it to train their models.

The raw tox21 dataset contains the SMILES of all compounds and labels for all 12 targets.

It is up to the users and researchers to collect the chemical information of each compound and decide which features are relevant to include for their model. As described in chapter 2.1.2, graph convolution models require a feature vector for each atom in a molecule. To augment each entry with such molecular information requires more than basic chemistry knowledge and falls outside the scope of this project. However, the application of graph convolution models to molecules is a recent field and there are currently no online chemistry databases that provide this kind of information. Hence, all feature data was acquired with DeepChem by using the graph convolution featurisation method which returns feature vectors of size 75 for any molecule. These features are listed in Table 4.1 below.

Feature	Description	Size
Atom type	43 Most common elements or other (one-hot or null)	44
Atom degree	Number of node connections 0-10 (one-hot or null)	11
Valency	Number of valence electrons 0-6 (one-hot or null)	7
Charge	Formal atom charge	2
Hybridization	sp, sp^2, sp^3, sp^3d or sp^3d^2 (one-hot or null)	5
Aromaticity	Whether the atom is part of an aromatic system	1
Hydrogen bonds	Number of Hydrogen atom bonded to the atom	5

Table 4.1: Features used to describe the chemical information of each atom in a molecule to create feature vectors of length 75.

Since the computational cost of the matrix multiplication between the adjacency matrix and the feature matrix is in the order of $O(N^2)$, it was decided to reduce the size of the dataset to 1,000 samples during the development of the model to allow for faster training time and checking of results. Based on the same logic, the model was trained on a 2 classification task instead of the 12 available targets. Only once the model is working correctly would the full dataset be used. The data is split into 60% training set, 20% testing set and 20% validation set with an even spread over all classes for all sets.

4.2 Model

The work by Kipf et al. (2017) [29] is designed for graph-based semi-supervised learning for node-level classification. The experimental setup is closely followed to replicate the core architecture of the model which is then adapted to supervised learning for graph-level classification. The super-node method described earlier in chapter 3.2 was chosen over the convolution-pooling-gather combo for ease of implementation. The second technique could be added as separate layers to the model to compare results once the model became functional. A GCN $f(X, A)$ which takes an adjacency matrix A and a feature matrix X as inputs, of a graph $\mathcal{G}$ with N nodes, has the layer-wise propagation rule below following from the generalised convolution equation 3.6:

$$X^{l+1} = \sigma(\widetilde{D}^{-\frac{1}{2}}\, \widetilde{A}\, \widetilde{D}^{-\frac{1}{2}}\, X^{l}\, \theta^{l}) \tag{4.1}$$

For every convolution layer l, the same graph adjacency matrix $A \in \mathbb{R}^{(N+1)\times(N+1)}$ is passed along with the output of the previous layer, X^{l}, and a new set of layer-specific weights, θ^{l}, is trained. $\sigma()$ represents the activation function of the layer, with ReLUs chosen for the input and hidden layers. A softmax activation function is applied row-wise for the classification output defined as $softmax(x_i) = \frac{1}{\mathcal{Z}}\, exp(x_i)$, where $\mathcal{Z} = \sum_i exp(x_i)$. Since this is a classification problem, the loss function is evaluated by the cross entropy error over all the predictions. The model is a two layer GCN with equation 4.1 fitted to the model to take the form of equation 4.2 below, where $\widehat{A} = \widetilde{D}^{-\frac{1}{2}}\, \widetilde{A}\, \widetilde{D}^{-\frac{1}{2}}$.

$$Z = f(X, A) = softmax(\widehat{A}\; ReLu(\, \widehat{A}\, X\, \theta^{l_0})\, \theta^{l_1}) \tag{4.2}$$

According to the testing of Kipf et al. (2017)[29], adding more than two convolutional layers decrease the performance of the model so this was not investigated further. The hidden layer has 32 units with the network weights θ^{l} initialised with the Glorot (or Xavier) algorithm [35], which scales the initialisation based on the number of input and output units, and updated via gradient descent. The model is trained up to a maximum of 200 epochs with the Adam optimiser and a learning rate of 0.01. An early stopping of window size 5 reduces overfitting by evaluating the loss function on the validation set. Dropout and an *L2* regularisation term are implemented in the code in case the model overfits the training set but these were unused.

The model is implemented in TensorFlow [36] for efficient GPU-based calculations with sparse matrix multiplications. The model only requires two inputs per graph: an adjacency matrix and a feature matrix. Tensorflow does not support forward passing of multiple rank 2 sparse Tensors (2D sparse matrices) so the adjacency matrices must be combined into a sparse block-diagonal matrix as per Figure 4.1. To ensure the convolution operation is applied on the features of neighbouring nodes in their respective graphs, the feature matrices of all graphs are simply concatenated vertically.

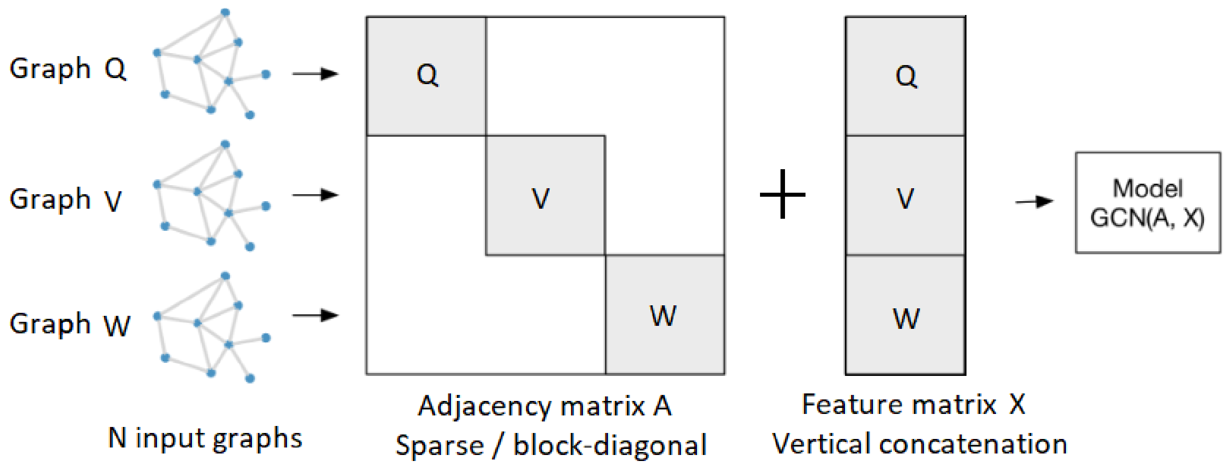


Figure 4.1: Pre-processing steps required to feed multiple graph instanced to the model: Each adjacency matrix for all input graphs are combined into a sparse, block diagonal matrix A. Each feature matrix for all input graphs are concatenated into a feature matrix X. Image modified from: [29]

4.2.1 Node-Level Classification to Graph-Level Classification

The architecture of the model is setup such that all operations are performed at node-level. This includes the classification step which is a result of applying a softmax activation function on the convolved feature matrices. The model is adapted to perform graph-level classification by using a bit-mask that informs the model which nodes are super-nodes. A bit mask is a boolean vector that defines which indexes are to be manipulated. In this case, the mask is passed to the loss and accuracy functions to ensure that the loss and accuracy are computed on the super-nodes only (for graph-level evaluation), and not any internal graph node.

As for the performance metric used, since this is a classification task, the area under curve (AUC) of the receiver operating characteristic (ROC) curve is evaluated.

4.3 Explaining Results With LIME

Despite the broad acceptance of machine learning models, they mostly remain black boxes. These predictive models are now used in many applications to help users make decisions. Users must first establish a level of trust with the model before letting it influence their choices. This is particularly important for applications involving health and safety such as drug design for this project. The most standard metric for evaluating the performance of a model is by accuracy of predictions on unseen data. However, such metrics can be misleading as data exclusive to the training set could be modeled, highlighting the importance of obtaining insight into the rational behind the model's predictions.

LIME (Local Interpretable Model-Agnostic Explanations), is an explanation software that aims to do exactly this. It can provide explanations to the predictions of a classifier for a single example "by learning an interpretable model locally around the prediction" [37]. It learns the behaviour of the classifier by altering the input and observing changes in the predictions. A dataset of perturbed instances is generated from the example being analysed by partitioning the features. The probability distribution of each instance belonging to the target class of the example is calculated according to the classifier along with the covariance between the instances. A linear, locally weighted learning model [38] is then trained on this data and outputs the highest weighted feature as an explanation to that particular example. Finally, users can use their intuition and business knowledge to draw conclusions on the reliability of the classifier. An illustration of how LIME works and how the explanations given can be interpreted is available in appendix B.

LIME has two requirements to be applicable to a model: models must output prediction probabilities which sum to 1 and all inputs must be of equal size. Unfortunately this means that LIME is not compatible with GCNs which operate on multiple graphs of variable sizes since the input is dependent on the number of nodes in a graph. For LIME to be compatible with GCNs, the data would have to be either multiple graphs with equal number of nodes, or one large network such that the data consists of only one graph, with every node having a feature vector of fixed length. This problem can be circumvented by reverting to using ECFPs, which encodes all entries into structural fingerprint vectors of identical lengths, and training a simple feedforward NN on this data. Instead of highlighting which molecular features contribute to the toxicity of a molecule, LIME would identify molecular sub-structures with the ECFP method. It is possible to bridge the gap between both models by using the toxic fingerprints determined by LIME on the ECFP model as input to the GCN, which should classify the fingerprint as toxic.

The task of classifying molecules in the Tox21 dataset as toxic or not, is repeated with the ECFP method. Each molecule is encoded as a structural fingerprint of length 1024 generated with RDKit to try and capture as many molecular fragments as possible. The feedforward NN used has the parameters recommended by MoleculeNet [31]. It consists of 1 hidden layer of 1,000 units with ReLU activation functions. The weights include a bias term and are randomly initialised and trained by gradient descent incorporating dropout. Finally, since it is a classification task, a softmax cross-entropy function is used to calculate the loss.

Chapter 5

Discussion

5.1 Results

The theory and design implementations of the literature was closely followed to develop a GCN. Unfortunately, the model does not behave correctly and outputs incorrect and inconsistent results. Consequently, the hyperparameters could not be optmised to the dataset and their effects on the model could not be investigated. Although, the model's performance is irrelevant to compare with the findings of other papers, these can still be briefly discussed. This section will attempt to discern the cause of the problem and provide an explanation to the results obtained. Regardless of the model's flaws, it remains possible to have an insight into what causes a molecule to be toxic by using LIME to interpret the results of a feedforward NN with the molecules encoded as ECFPs.

5.1.1 Model Analysis

The graph Laplacian operator is unique to each molecule since it is a representation of the connections between the nodes of the molecular graph. The topological structure of the graph remains unchanged throughout the model regardless of the layer depth, allowing for the layer-wise propagation rule, equation 4.1, to be formulated from the generalised convolution equation 3.6. These graph Laplacians can be visualised, three examples are displayed in Figure 5.1 with the super-node as the last Laplacian matrix index. A large molecule is included in the examples to highlight the sparsity of the graph Laplacian. A single convolution operation has been carefully followed step by step to confirm that the data is formatted correctly.

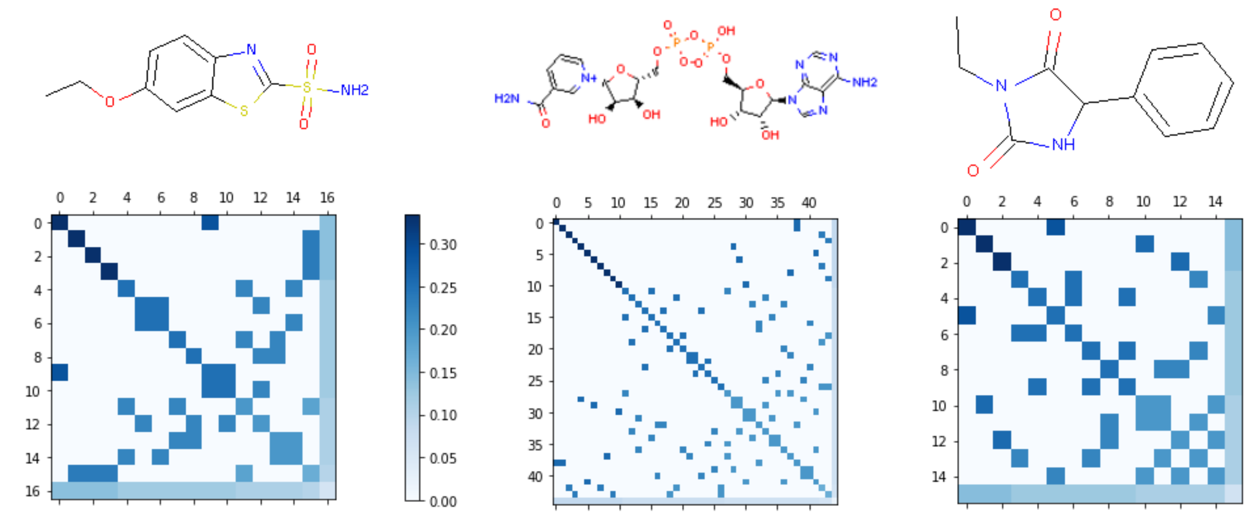


Figure 5.1: Three examples of unique graph Laplacian operators generated from the molecules above them. The colour map represents the normalised node degree value.

The outputs and performance of the model is analysed by examining the training and testing accuracies, the training and validation losses, as well as the ROC curve for the two classes.

The accuracy and loss plots of machine learning models tend to follow a general shape, so an expectation of the results can be formed. The typical accuracy of a model is expected to start low, for both the training and the testing sets, on the first epoch and both should gradually increase before slowly converging to a value. The accuracy of the training set is expected to be higher than the testing set, and different runs should yield very similar results. The opposite trend should be observed for the loss function. The loss of the training and validation sets should start high and slowly decrease over multiple epochs. The loss on the training set is expected to be lower than the loss on the validation set. The model should then stop once the validation loss begins increasing, preventing overfitting of the training set.

The accuracy plots, two examples are shown in Figure 5.2, generated do not match the expectation but are rather unusual and difficult to interpret since they are inconsistent between runs. The accuracies of both the training and the testing sets on the first epoch are different between each run and fall in the rather large range between 25% and 75%. These accuracies sometimes increase or decrease but both follow the same trend, with the final values lying within a 10% margin of the 50% mark, but neither converges. Such a high starting accuracy could mean that the model already has a prior knowledge of the results as if the labels were included in the training data. However the labels are clearly separated from each other and if this was the case, the model's accuracy would rapidly increase. A high starting accuracy on the test set could entail that some of the testing data is included in the training set but these two are separate partitions of the original data. This leads to one of two possibilities: the convolution operations are not performed correctly or the accuracy is not calculated correctly. The former has been thoroughly checked and disproven so the error must lie in the calculation of the accuracy. However, the calculation of the accuracy is done via an inbuilt function, so the source of the problems must originate from the values passed to the function. The logical conclusion to draw is that the bit mask described in section 4.2.1 is not functioning properly resulting in errors in the accuracy function.

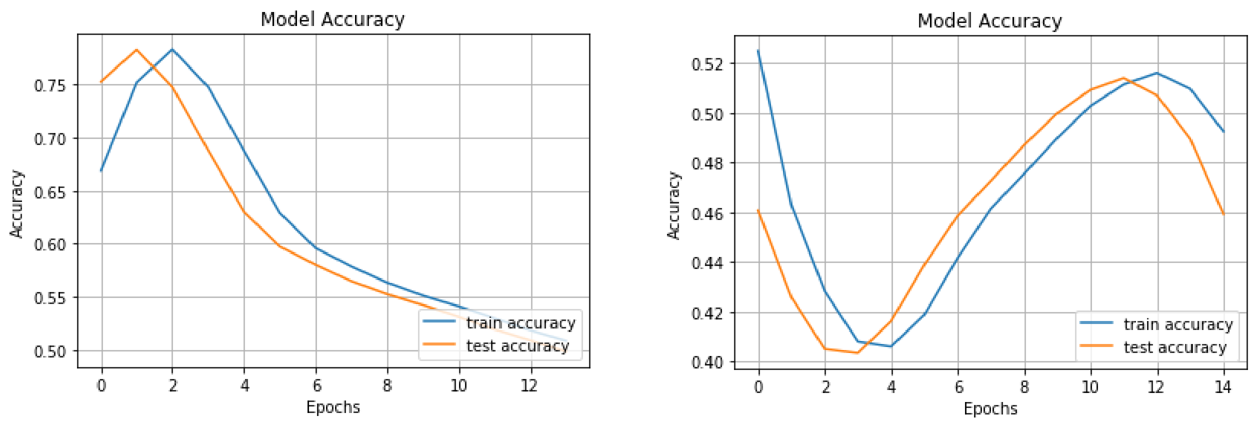


Figure 5.2: Training and testing accuracies obtained from two separate runs of the model.

Looking at the training and validation losses in Figure 5.3, the general trend seems to be reasonable and the same plot is produced each run. The loss values start high and slowly decrease with the model terminating once the validation loss begins increasing so the early stopping mechanism works as expected. However a discrepancy of the expectation can be observed. Although the losses are decreasing over the number of epochs, the change in value of the losses seem to be minimal which implies that the model is not learning properly. Since the bit mask is passed to the loss function in exactly the same way as for the accuracy function, which has been identified to be erroneous, it is likely that these values are actually incorrect and do not reflect the loss of the graph classification.

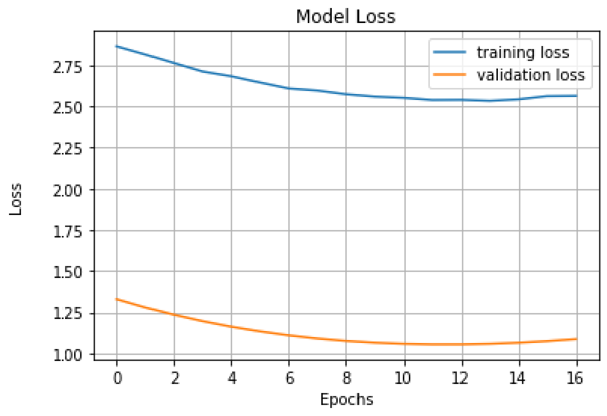


Figure 5.3: Training and validation losses

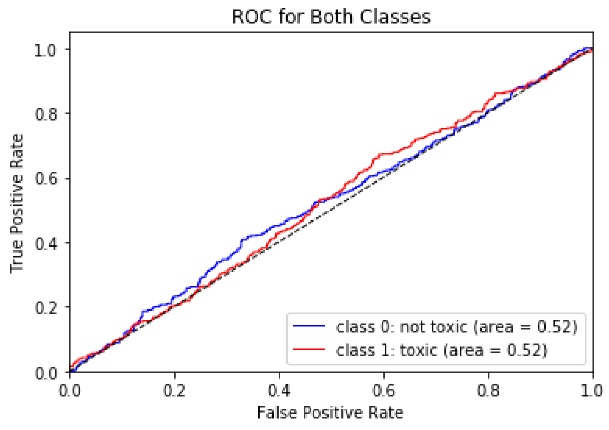


Figure 5.4: ROC for both classes

Any classification task results are best interpreted by inspecting the ROC and calculating the ROC-AUC, shown in Figure 5.4. Alas, the poor results of the model leave little room to discuss. The ROC of both classes essentially follow the diagonal reference line so the model is as performant as a random classifier.

The performance of GCNs can still be briefly discussed by comparing the findings of papers that applied their models to the Tox21 dataset. Their results are presented in Table 5.1 using ROC-AUC as the evaluation metric, and includes the ECFP model for reference. It is interesting to note that whilst the table is ordered by ROC-AUC score, it also happens to be ordered by date which is evidence of the GCN architecture improving over the years. It was mentioned at the beginning of this dissertation that GCNs were expected to surpass the ECFP technique by encoding the full connectivity of the molecule instead of its sub-structures. The ECFP technique outperforms most GCN architectures but it is a more recent and improved model. The same team which designed the ECFP model (MoleculeNet [31]), were able to obtain considerably better results with their GCN model, confirming that GCNs have less information loss than ECFP models.

Model	Validation	Testing
GCN [31]	0.825	0.829
ECFP [31]	0.755	0.788
[23]'s GCN architecture [39]	0.754	0.748
Neural Fingerprint GCN [30]	0.750	0.734
[22]'s spectral GCN achitecture [39]	0.711	0.702
This model	0.52	0.52

Table 5.1: Comparison of average ROC-AUC scores on the Tox21 dataset from papers having implemented GCNs.

5.1.2 Identifying Toxic Molecular Fragments

Section 4.3 introduced the importance of establishing trust in a model if it were to be used in scenarios requiring critical decision taking. NNs are trained on a dataset with the objective of modeling relationships between inputs and outputs and learn statistical patterns to apply these learned rules on new, unseen data. The reasoning behind it is that the machine can learn a general rule for a certain task given enough training examples. The success of these powerful predictive models has been recognised and they are now being implemented in a huge range of applications from computer vision related task, market predictions, fraud detections, and even medical and legal advice.

The issue is that no explanation is provided as to which factors and features influenced the prediction output. Evaluating the prediction accuracy on the test set is the most common metric for measuring the model's performance. This is a good indicator but remains a small sample size compared to real-world data. ROC and the calculation of the AUC is a more precise measure for classifiers but the problem remains. The risk of the model identifying correlations between variables which are unique to the dataset is still there and it can be challenging to realise this error. Algorithms such as feature selection and Principle Component Analysis (PCA) seek to identify irrelevant and redundant features to reduce the number of attributes and decrease the complexity of the model. These can also serve as indicators to discern the more important and most influential features regarding the global behaviour of the model. This is usually sufficient to build confidence in the model's performance but not substantial enough to trust the model with decision which could harm the health and safety of people. For example, doctors need to know which symptoms led to a patient's disease and chemists require some insight into why a drug is toxic and dangerous to a person's health. As explained in section 4.3, LIME is an explanation software which seeks to present these explications.

As per chapter 4.3, the task of classifying molecules in the Tox21 dataset as toxic or not, is repeated with the ECFP method using structural fingerprint of length 1024. Since LIME provides explanations for individual instances and not a global explanation for how the model generally behaves, the successive paragraphs follow the example for the first molecule classified as toxic, depicted in Figure 5.5. The LIME explanation for that particular molecule is shown in Figure 5.6.

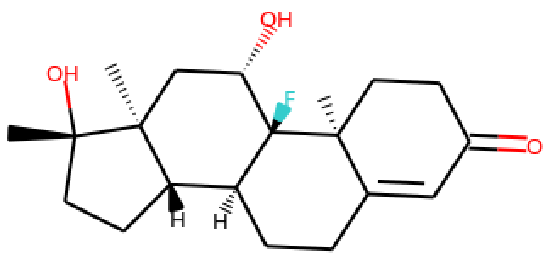


Figure 5.5: First molecule to be correctly identified as toxic by the model

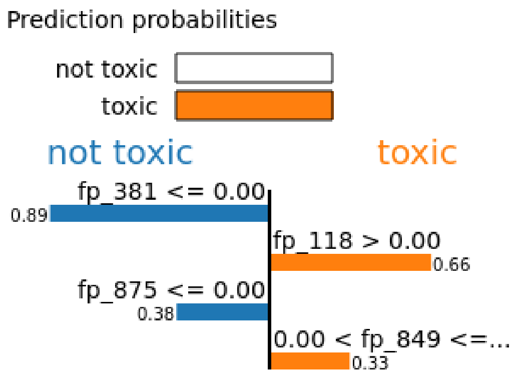
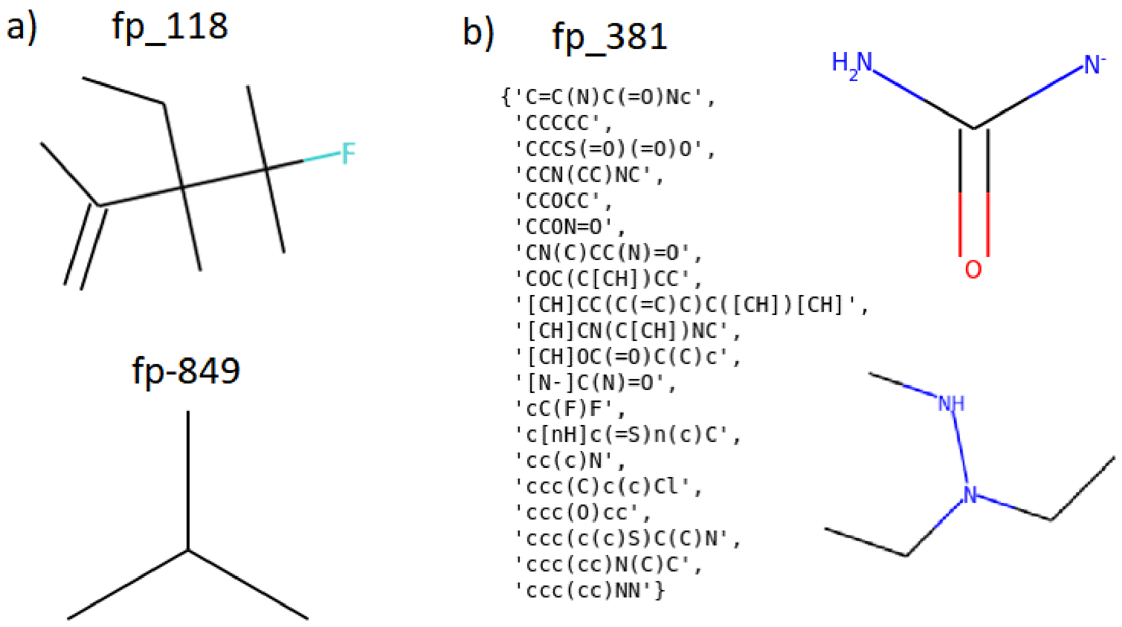


Figure 5.6: Output explanation by LIME regarding the molecule on the left side of this figure.

Figure 5.6 can be counter-intuitive to understand so it will be presented in detail. The top half of the image states the classification result of the molecule: it is toxic. The bottom half displays the features, in this case molecular fingerprints, which most influenced the decision of the model when classifying the molecule as toxic or not. According to the explanation, fingerprints #118 and #849 are the molecular fragments within the molecule which contribute the most to the molecule's toxicity. Before analysing the features contributing to the molecule's non-toxicity, it is important to understand that some elements such as Fluorine are known to be more toxic than others and tend to me the leading cause for toxic properties in compounds. However, chemical properties vary with the composition and connectivity of compounds so this is not always the case. This means that the search space of toxic fragments is much smaller than the search space for non-toxic fragments [40]. It is therefore more relevant for the explainer to identify fingerprints which are likely to increase the molecule's toxicity, than list all the non-toxic fingerprints (which constitutes the majority of them). On this note, fingerprints #381 and #875 are actually lists of fingerprints. These fingerprints are not present in the molecule but would increase the molecule's toxicity levels if they were part of it. Hence, fingerprint lists #381 and #875 contribute to the molecule not being toxic because the fingerprints within the lists are absent in the molecule.

Once the explainer class can be interpreted, the fingerprints activated by the model can be fetched and visualised with RDKit. Figure 5.7 presents the fingerprints #118 and #849 which are believed to be toxic, as well as the list of fingerprints #381 with two example fragments. At this stage, a field expert is required to analyse these molecular fragments. With their experience and knowledge, they can form a basis on whether the model is reliable and trustworthy. In order to confirm the toxicity of these molecular fragments and to link these findings to the graph convolution model, the fingerprints can be treated as molecules and encoded in the same manner as the data from the Tox21 data to use as input into the graph convolution model. The GCN would classify the small molecular structure as toxic or non-toxic, thus validating or disproving the explanation provided by LIME and further establishing the reliability of the model. Unfortunately, this final step cannot be tested since the GCN model does not behave properly. The importance of such insights into a model's decisions remains crucial for applications related to human safety, such as the drug design process, if regulations were ever to acknowledge these predictive models as useful.



```
{'C=C(N)C(=O)Nc',
 'CCCCC',
 'CCCS(=O)(=O)O',
 'CCN(CC)NC',
 'CCOCC',
 'CCON=O',
 'CN(C)CC(N)=O',
 'COC(C[CH])CC',
 '[CH]CC(C(=C)C)C([CH])[CH]',
 '[CH]CN(C[CH])NC',
 '[CH]OC(=O)C(C)c',
 '[N-]C(N)=O',
 'cC(F)F',
 'c[nH]c(=S)n(c)C',
 'cc(c)N',
 'ccc(C)c(c)Cl',
 'ccc(O)cc',
 'ccc(c(c)S)C(C)N',
 'ccc(cc)N(C)C',
 'ccc(cc)NN'}
```

Figure 5.7: a) fingerprints #118 and #849 identified by LIME as contributing the most to the molecule's toxicity. b) The list of fingerprints held in list #381 which are not present in the molecule and contributing to its non-toxicity with two rendered examples.

5.2 Limitations

Reflecting on the overall project, several limitations can be identified.

GCNs require all information to be represented by a feature vector for each node. The application of this type of model on molecules is a recent advancement and the DeepChem library is the only source found that provides this capability. Since none of the papers on this topic discussed which chemical information is important to include in feature vector, the default features returned by the DeepChem function were used. No customisation of features was considered in this project. If a chemist or field expert could provide some intuition as to which other features would be useful to include, the function can be modified to include these recommendations. For example, [32] which uses Weave modules, included whether the atom (node) was part of a ring structure and one-hot encoded the size of the ring.

Obviously, the analysis of the model was restricted to explaining why it produced the results presented since it does not function properly. The model was trained on a portion of the full dataset while it was under development. Only when the model is operative would it be trained on the full dataset to improve its performance with more training examples. Following the same reasoning, the model was reduced to work on a two class classification task instead of the 12 targets available in the dataset.

The LIME software, although powerful, requires inputs of same size and therefore is not compatible with the graph convolution model. To circumvent this limitation, LIME was applied to the ECFP method to gain some knowledge about what is responsible for the toxicity in molecules. This yielded interesting results but the GCN remains a black box model so there is no basis to evaluate any trust in this type of model.

Finally, no batch-wise training was implemented. Since only 1,000 molecules were used while devloping the model, the computational power required to run the model remained relatively low so it was not necessary to batch the data. However, the computational cost of the matrix multiplication between the adjacency matrix and the feature matrix is in the order of $O(N^2)$ so batching the data would be imperative. To deal with graphs of variable sizes, a row and column-wise zero padding would ensure the batch size remains consistent for the entire dataset.

5.3 Extensions and Future Work

Building on the limitations discussed in the previous section, a few extensions to the project can be suggested. The gaps identified in the literature and the ongoing development of the LIME software sets the scene for future works.

Scarselli et al. (2011) [19] and Kipf et al. (2017) [29] investigated the theory of creating a dummy super-node to classify graphs in the same manner that node-classification is carried out. However, no research has been conducted on the difference in model performance between using super-nodes and implementing a graph-pooling operation after each convolution followed by a graph gather operation at the end of the network. This project can easily be further developed for the model to incorporate both techniques. The effects can be analysed by comparing the results obtained and finally clarify which technique is more performant. Pooling operations update the graph features with a higher level of representation, a technique proved to be effective with CNNs. However, some of the intuition is lost on graphs which are not reduced in size (via graph coarsening) since repeated use of pooling operations on small graphs (such as molecules) would lead to all the nodes sharing the same feature vector. The reason being that the feature values are shared between adjacent nodes and propagated by one-hop per pooling operation. This might also depend on the structure and connectivity of the atoms but these remain theories to be tested.

Additionally, the explanations given by LIME were presented for a single example. If this was to be carried out for all molecules in the dataset, an opportunity is created to compare the outputs for all instances and identify re-occuring toxic fingerprints. This would provide a deeper insight into the leading cause of toxicity in molecules and would contribute to further evaluating the reliability of the model.

Finally, it would be extremely interesting if advancements could be made on the LIME software to allow it to read inputs of different sizes. This would render the software to be compatible with GCNs for more valuable explanations to be extracted. Instead of identifying which molecular fingerprints are toxic, the explainer function would list the exact features the model believes contribute to the molecule's toxicity. This sort of information is much more precise and would allow for more conclusive results to be drawn. Some of the contributors of DeepChem have mentioned tackling earlier this year but with no further updates [41].

Chapter 6

Conclusion

The performance of machine learning models has been steadily improving over the years with novel model architectures and with the help of more powerful graphics processing units. Particular attention has been devoted to Convolutional Neural Networks due the versatility of their applications in multiple domains and to leverage their powerful ability to solve high dimensional learning problems. Convolutional Neural Networks have proved to obtain state-of-the art results when applied to audio, images as well as videos, and are now being extended to work on graphs which can represent any data which does not lie in the Euclidean domain. Research which led to the design of Graph Convolutional Networks (GCNs).

The incentives of applying convolution on graphs have been clearly stated, specifically for the application to molecules in field of chemoinformatics, and why GCNs were believed to surpass the previous state-of-the-art molecular encoding technique: Extended-Connectivity Fingerprints (ECFPs). Additionally, the importance of providing an explanation to a model's predictions was stressed in order to establish a level of trust and confidence in the model. Users require insight into the model's decision making process if such models were ever to adopted in applications involving human health and safety such as in the drug discovery process.

The aim of this dissertation focused on clarifying the inner workings of GCNs and explain the theory behind graph convolution. The first objective was completed by presenting a detailed analysis of the literature, which generally studies the problem from either a spatial or spectral approach. From the research conducted, a methodology was tailored to the task of developing a GCN to classify molecules in the Tox21 dataset, as toxic or non-toxic, using the super-node technique. The model did not behave properly but it was still demonstrated through a ECFP-based Neural Network how the predictions of a model can be explained with the powerful interpretative capabilities of the LIME software, completing the remaining project objectives.

Finally, the limitations of the project were identified and extensions were proposed for future work. The most promising future project is to investigate the difference in performance between between a GCN model which uses the super-node method as opposed to implementing a graph pooling operations after each convolution as this is gap identified in the literature.

References

[1] J. Gu, Z. Wang, J. Kuen, L. Ma, A. Shahroudy, B. Shuai, T. Liu, X. Wang, G. Wang, J. Cai, and T. Chen, “Recent advances in convolutional neural networks,” *Pattern Recognition*, vol. 77, pp. 354–377, 2017.

[2] J. Gasteiger, “Chemoinformatics: Achievements and challenges, a personal view,” *Molecules*, vol. 21, pp. 151–166, 2016.

[3] T. Engel, “Basic overview of chemoinformatics,” *Chemical Information and Modeling*, vol. 46, pp. 2267–2277, 2009.

[4] N. C. for Advancing Translational Sciences, “Toxicity of the 21st century.” https://ncats.nih.gov/tox21, Accessed: Aug-2018.

[5] K. Liu, X. San, L. Jia, J. Ma, H. Xing, J. Wu, H. Gao, Y. Sun, F. Boulnois, and J. Fan, “Chemi-net: A molecular graph convolutional network for accurate drug property prediction,” *CoRR*, 2018.

[6] P. Willet, “Special issue: Chemoinformatics,” *Molecules*, vol. 21, pp. 535–538, 2016.

[7] R. M. Hanson, “Jmol smiles and jmol smarts: specifications and applications,” *Journal of Chemoinfomatics*, vol. 8, 2016.

[8] Daylight Chemical Information Systems, *Daylight Theory Manual*, 4.9 ed., 2011.

[9] “Rdkit: Open-source cheminformatics and machine learning.” http://www.rdkit.org, Accessed: Aug-2018.

[10] D. Rogers and M. Hahn, “Extended-connectivity fingerprints,” *Chemical Information and Modeling*, vol. 50, pp. 742–754, 2010.

[11] A. Cereto-Massague, M. J. Ojeda, C. Valls, M. Mulero, S. Garcia-Vallve, and G. Pujadas, “Molecular fingerprint similarity search in virtual screening,” *Methods*, vol. 71, pp. 58–63, 2015.

[12] W. Torng and R. B. Altman, “3d deep convolutional neural networks for amino acid environment similarity analysis,” *BMC Bioinformatics*, vol. 18, 2017.

[13] I. Wallach, M. Dzamba, and A. Heifets, "Atomnet: A deep convolutional neural network for bioactivity preiction in strucure-based drug discovery," *Computing Reseach Repository*, 2015.

[14] Authors of MoleculeNet and 57 contributors, "Deepchem: Democratizing deep-learning for drug discovery, quantum chemistry, materials science and biology." https://tripod.nih.gov/tox21/challenge, Accessed: Aug-2018.

[15] P. Bezak, P. Bozek, and Y. Nikitin, "Advanced robotic grasping system using deep learning," *Procedia Engineering*, vol. 96, pp. 10–20, 2014.

[16] X. Zhu, J. Lafferty, and Z. Ghahramani, *Combining Active Learning and Semi-Supervised Learning Using Gaussian Fields and Harmonic Functions.* 2003.

[17] O. Chapelle and A. Zien, "Semi-supervised classification by low density separation," in *AISTATS*, pp. 57–64, 2005.

[18] J. Weston, F. Ratle, and R. Collobert, "Deep learning via semi-supervised embedding," *Proceedings of the 25th international conference on Machine Learning*, pp. 1168–1175, 2008.

[19] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, "The graph neural network model," *IEEE Transactions on Neural Networks*, vol. 20, pp. 61–80, 2009.

[20] Q. Li, Z. Han, and X. M. Wu, "Deeper insights into graph convolutional networks for semi-supervised learning," *Computing Reseach Repository*, 2018.

[21] H. Altae-Tran, B. Ramsundar, A. S. Pappu, and V. Pande, "Low data drug discovery with one-shot learning," *ACS Central Science*, vol. 3, pp. 283–293, 2017.

[22] J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun, "Spectral networks and deep locally connected networks on graphs," *arXiv:1312.6203*, 2014.

[23] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast loalized spectral filtering," *arXiv:1606.09375v3*, 2017.

[24] X. Bresson and T. Laurent, "Residual gated graph convnets," *Computing Reseach Repository*, 2018.

[25] T. N. Bui and C. Jones, "Finding good approximate vertex and edge partitions is np-hard," *Information Processing Letters*, vol. 42, pp. 153–159, 1992.

[26] J. Shi and J. Malik, "Normalized cuts and image segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, pp. 888–905, 2000.

[27] S. Mallat, *A wavelet tour of signal processing: The Sparse Way.* Academic Press, 1998.

[28] D. K. Hammond, P. Vandergheynst, and R. Gribonval, "Wavelets on graphs via spectral graph theory," *Applied and Computational Harmonic Analysis*, vol. 30, pp. 129–150, 2011.

[29] T. N. Kipf and M. Welling, "Semi-supervised classificatin with graph convolutional networks," in *International Conference on Learning Representations (ICLR)*, 2017.

[30] D. Duvenaud, D. Maclaurin, J. Aguilera-Iparraguirre, R. Gomez-Bombarelli, T. Hirzel, A. Aspuru-Guzik, and R. P. Adams, "Convolutional networks on graphs for learning molecular fingerprints," *arXiv:1509.09292v2*, 2015.

[31] Z. Wu, B. Ramsundar, E. N. Feinberg, J. Gomes, C. Geniesse, A. S. Pappu, K. Leswing, and V. Pande, "Moleculenet: A benchmark for molecular machine learning," *Chemical Science*, vol. 9, pp. 513–530, 2017.

[32] S. Kearnes, K. McCloskey, M. Berndl, V. Pande, and P. Riley, "Molecular graph convolution: moving beyong fingerprints," *Journal of Computer-Aided Molecular Design*, vol. 30, pp. 595–608, 2016.

[33] Y. Li, D. Tarlow, M. Brockschmidth, and R. Zemel, "Gated graph sequence neural networks," *arXiv:1511.05493v4*, 2017.

[34] "2014 tox21 challenge." https://tripod.nih.gov/tox21/challenge, Accessed: Aug-2018.

[35] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep forward neural networks," in *International Conference on Artificial Intelligence and Statistics (AISTATS)*, vol. 9, 2010.

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

[37] M. T. Ribeiro, S. Singh, and C. Guestrin, "'why should i trust you?' explaining the predictions of any classifier," *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1135–1144, 2016.

[38] C. G. Atkeson, S. A. Schaal, and A. W. Moore, "Locally weighted learning," *AI Review*, vol. 11, pp. 11–73, 1997.

[39] R. Li, S. Wang, F. Zhu, and Huang, "Adaptive graph convolutional neural network," *arXiv:1801.03226v1*, 2018.

[40] Hazard Evaluation System and Information Service (HESIS), *Understanding Toxic Substances: An Introduction to Chemical Hazards in the Workplace*, 2018.

[41] "Deepchem issue raised # 1184." https://github.com/deepchem/deepchem/pull/1184, March 2018.

Appendix

Appendix A
An example of multiresolution clustering of a graph presented in [22].

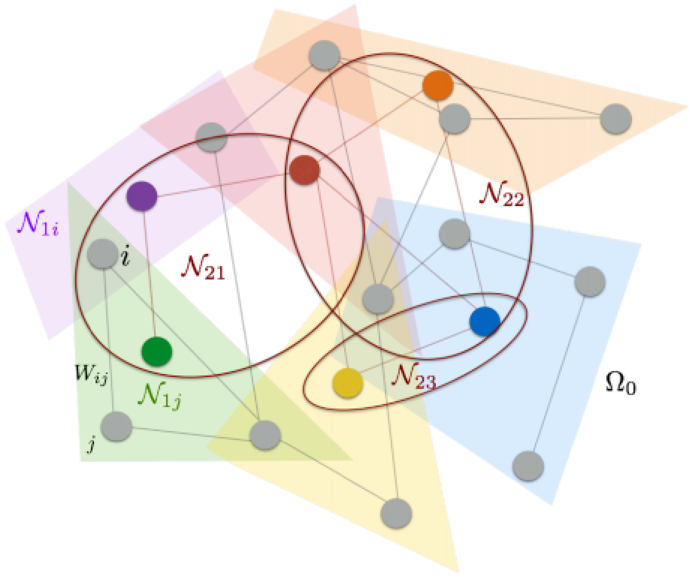


Figure 6.1: Example of a multiresolution clustering of an undirected graph with two levels of clustering. The original graph is depicted by grey nodes and the new graph is represented by the coloured nodes. [22]

Appendix B

Figure 6.2 is directly taken from the paper, [37], and demonstrates how the explanations given by the software should be interpreted.

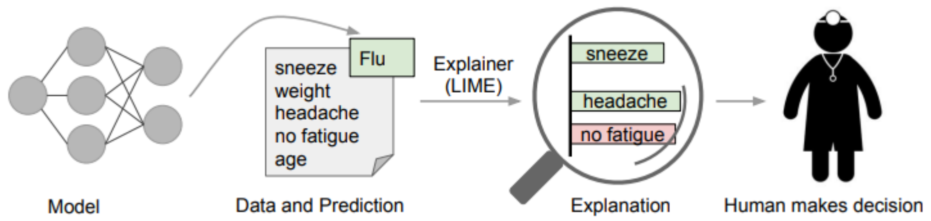


Figure 1: Explaining individual predictions. A model predicts that a patient has the flu, and LIME highlights which symptoms in the patient's history led to the prediction. Sneeze and headache are portrayed as contributing to the "flu" prediction, while "no fatigue" is evidence against it. With these, a doctor can make an informed decision about the model's prediction.

Figure 6.2: LIME overview [37]

Figure 6.2 illustrates how LIME compiles the explanations by analysis the model for the example of classifying an image as a tree frog.

Source: M. T. Ribeiro, S. Singh and C. Guestrin, "Introduction to Local Interpretable Model-Agnostic Explanations (LIME)", 2016, Accessed on August 2018 at: https://www.oreilly.com/learning/intr
to-local-interpretable-model-agnostic-explanations-lime

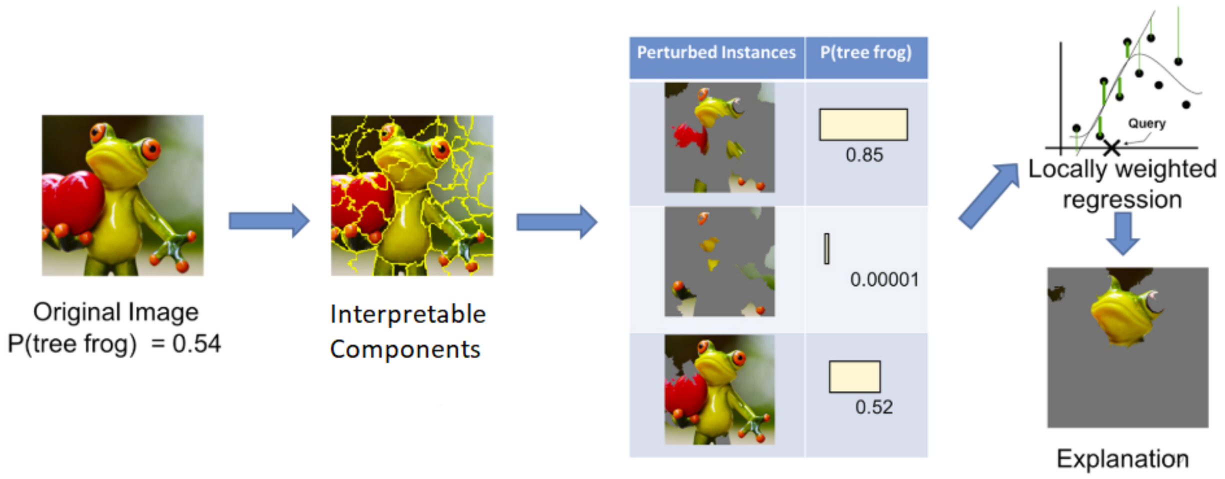


Figure 6.3: Visualising how LIME works with an image as worked example. Image modified from the source provided.