MASTERARBEIT / MASTER'S THESIS

Titel der Masterarbeit / Title of the Master's Thesis

Anomaly detection in chemical space

verfasst von / submitted by

Matthias Welsch, BSc

angestrebter akademischer Grad / in partial fulfilment of the requirements for the degree of

Master of Science (MSc)

Wien, 2023 / Vienna, 2023

Studienkennzahl lt. Studienblatt / degree programme code as it appears on the student record sheet:	A 066 910
Studienrichtung lt. Studienblatt / degree programme as it appears on the student record sheet:	Computational Science
Betreut von / Supervisor:	Ass.-Prof. Dipl.-Inf. Dr. Nils Morten Kriege

Abstract

Whilst classification of chemical compounds is a well-established field, there have been few works on anomaly detection, and many do not explicitly specify what a chemical anomaly is. In general, there are four key problems in anomaly detection: unknowness, unlikeness, rarity, and irregularity, but many definitions of chemical anomalies do not tackle all of these challenges. The four problems of anomaly detection are formalized and based on these issues benchmark anomaly detection tasks are compiled. Furthermore, the datasets are analysed by performing classification experiments to ensure that algorithms are able establish a decision boundary between inliers and anomalies. For balanced training sets, classification does not pose a problem. Anomaly detection techniques based on different paradigms are selected and the datasets are tested. These algorithms are unable to correctly identify anomalies. A novel anomaly detection technique based on frequent pattern mining is devised that is able to solve the tasks. This shows that the datasets are feasible in an anomaly detection setting, if the correct features are selected.

Zusammenfassung

Während Klassifikation von Chemikalien ein wohletabliertes Feld ist, gibt es wenige Arbeiten zu Anomaliedetektion und viele davon beschäftigen sich nicht damit wie eine Anomalie aussieht. Es gibt vier zentrale Probleme in der Anomaliedetektion: Unbekanntheit, Unähnlichkeit, Seltenheit, und Unregelmäßigkeit, aber viele Definitionen für Anomalie beschäftigen sich nicht mit all diesen Problemen. Diese Arbeit formalisiert diese vier Probleme und erstellt chemische Datensätze basierend auf diesen Herausforderungen. Die Datensätze werden mit Klassifikationsverfahren getestet, um zu zeigen, dass es Algorithmen gibt, die eine Entscheidungsgrenze zwischen normalen und anormalen Datenpunkten ziehen können. Es werden Anomaliedetektiontechniken, die auf verschiedenen Paradigmen beruhen, mit den erstellten Datensätzen getestet, allerdings schaffen es diese Algorithmen nicht Anomalien korrekt zu identifizieren. Eine neue Anomaliedetektionsmethode wird konstruiert, die in der Lage ist die erstellten Datensätze zu lösen. Dies zeigt, dass die Datensätze in einem Anomaliedetektionskontext verwendbar sind, wenn der Algorithmus die richtigen Merkmale auswählt.

Contents

1 | Introduction

In recent years, the interest in anomaly detection has increased as there are many applications in security, AI safety, and health and medical risk. A transaction graph changing in an unexpected way, a masked person entering a building by destroying a window at night on surveillance footage and a diabetic's blood sugar levels dropping below a certain threshold are examples of anomalous events that indicate problems. These events are rare but also highly interesting [1]. In chemistry anomaly detection has important applications in predictive modeling and drug discovery [2]. These include but are not limited to:

- Characterizing the applicability domain of quantitative structure-activity relationships [3]
- Rectifying data sets [2]
- Screening virtual libraries [4]

In Figure 1.1 some of the applications of anomaly detection in chemistry are displayed. In sub-figure (a), it is evident that the model based on the green data points is not applicable to the red data point because it is far away. In QSAR modelling the structure - chemical property relationship is described by a mathematical model. This kind of modelling is essential in order to reduce the amount of animal experiments that have to be performed because the number of chemicals that have to be tested can be narrowed down by in silico experiments. For validating such models, the definition of an applicability domain is required because QSAR models are reductionistic and hence have limitations on their prediction power when it comes to chemical structures that are far away from the molecules that the model was built upon [3].

In sub-figure (b) the fitting of the curve is heavily skewed by the presence of the red data point in the training set. In previous approaches anomalies were detected by looking at the prediction error of the model and after rectification the model was retrained in order to improve the reliability [2]. A problem with this approach is that the anomalous behavior is defined with respect to the

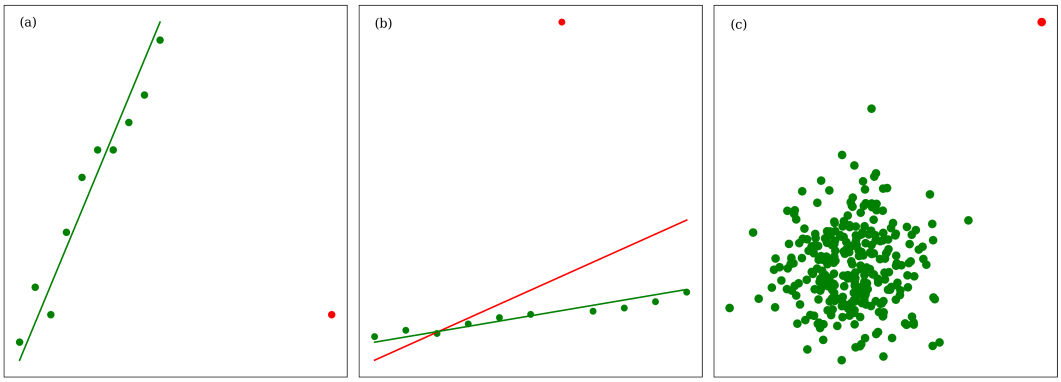

Figure 1.1: (a): Characterization of applicability domain, (b): Anomalous point leading to a bad fit, (c): red point is different from the rest

model and hence it is possible that the prediction error of a molecule is small, but the overall fit of the model suffers.

In sub-figure (c) the red point is different from the rest and therefore has to be treated differently. In virtual library screening anomaly detection techniques could be used as tools to improve the drug discovery process. For example, compounds that are dissimilar to the regular compounds, can be found and then either be treated differently than the rest or discarded for further analysis [4].

Whilst it is easy to remove anomalies based on a certain chemical property [5] or based on the prediction error [2], there is little research when it comes to detecting anomalies a priori. This thesis will focus on creating new test sets for anomaly detection, show that these test sets are feasible with regular classification approaches, and show that while solving the anomaly detection problem for these sets is possible, non-tailor-made state of the art anomaly detection techniques yield bad results.

1.1 General problems in anomaly detection

Let $\mathcal{X} = \{x_1, \ldots, x_N\}$ be the examples with x_i being the instance i and with the label $y_i \in \mathcal{Y} = \{0, 1\}$ for all $i \in \{1, ..., N\}$, where 0 indicates normality and 1 indicates abnormality. Further, let $\mathcal{I} = \{x_i \in \mathcal{X} \mid y_i = 0\}$ be the set of all inliers and $\mathcal{A} = \{x_i \in \mathcal{X} \mid y_i = 1\}$ be the set of all anomalies. It is assumed that there is a non-trivial property $\mathcal{P}$ such that $\mathcal{P}(a) = \mathcal{P}(b)$ for all $a, b \in \mathcal{I}$ and $\mathcal{P}(a) \neq \mathcal{P}(b)$ for all $a \in \mathcal{I}, b \in \mathcal{A}$ because if there were no property that indicates normality - with other words something that the inliers have in common whilst the anomalies behave differently in that regard, the notion of abnormality does not make sense. Therefore, there is also a property $\hat{\mathcal{P}}$ such that $\hat{\mathcal{P}}(x_i) = y_i$. The goal of anomaly detection is to approximate $\hat{\mathcal{P}}$ with an algorithm $\mathcal{F}$. $\mathcal{F}$ can be a traditional algorithm as well as a machine-learning one. In order to train $\mathcal{F}$ only a small amount of data instances is available, and it is even possible that there is no known anomaly at all.

When doing anomaly detection (also known as outlier detection, novelty detection or one-class classification) there are four distinct problems:

- Unknowness: It is unknown how an anomaly looks like until it actually occurs [1]. In other words: The result of $\mathcal{F}$ should not change if additional anomalies become available for training and this is why the algorithm should not be given labeled anomalies as training examples because otherwise $\mathcal{F}$ is given access to additional knowledge about $\mathcal{A}$ which causes it to change based on examples in $\mathcal{A}$. Hence $\widetilde{\mathcal{I}} \subseteq \mathcal{I}$ should be used as a training set in supervised algorithms, otherwise it is likely a classification problem that focuses on finding the boundary between two classes and not the boundary around one class.

- Unlikeness: Outliers are irregular and therefore two outliers not necessarily have similar properties [1]. As given by the definition the anomalies can be identified by the property $\hat{\mathcal{P}}$, but there might not be any other non-trivial properties that can identify anomalies.

- Rarity: Most of the data instances are not anomalous and therefore it is impossible to collect a large number of novelties as training data [1]. E.g., $|\mathcal{I}| \gg |\mathcal{A}|$. Rarity combined with unlikeness is the reason why training $\mathcal{F}$ on all instances in an unsupervised setting should be fine because the influence of the small number of anomalies on the boundary around the training set should be negligible.

- Irregularity: There are three distinct types of anomalies:

  - Point anomaly: A single data instance is anomalous with respect to the majority of the data

- Group anomaly: There is a small group of data points that are similar to each other but all of them are dissimilar to the rest of the data [1]. In some cases there exists a non-trivial subset $\widetilde{\mathcal{A}}$ of $\mathcal{A}$ that shares a non-trivial property $\mathcal{P}'$ that is different from $\hat{\mathcal{P}}$ that can differentiate $\widetilde{\mathcal{A}}$ from $\mathcal{I}$ but that is not necessarily the case.
- Conditional anomaly: Data instances are only considered anomalous in a specific context [6]. $\mathcal{P}$ can change based on additional information about the inliers.

For example, when imagining a dataset consisting of 2000 peptides, if an alloy and C_{60} were added to that set, the expected number of anomalies would be two. Before adding these two data instances it is unknown how the anomalies might look like (Unknowness) except by the dissimilarity to the inliers. The inorganic compounds are not similar in any way (Unlikeness), and they occur only around 0.1% of the time (Rarity). If ten additional alloys were added, all alloys are a group anomaly whilst the Buckyball is a point anomaly. If there is the additional information that the peptides of the original set should be found in humans and there are occurrences of non-human peptides, then these would be considered conditional anomalies and likely group anomalies as well.

These problems lead to six reoccurring challenges:

- Low recall: Identifying all anomalies is still challenging as they are unknown, but it would be very important to get most anomalies correct as these are the interesting data points for example for virtual library screening [7].
- High-dimensional data and interdependency: It is difficult to detect anomalies in high-dimensional data because the anomalous properties might be hidden in a low-dimensional subspace but it is also possible that high-order, heterogeneous, and nonlinear coupling is needed in order to detect anomalies [8] or that there are interdependencies in the data as in node-level anomaly detection [9].
- Data-efficient learning: In many cases there are no large, labeled sets for supervised anomaly detection and using unsupervised techniques relies on assumptions about the distributions.
- Noise: Noise can lead to incorrect classification of a data point as an anomaly.
- Heterogeneous data sources: There are multiple types of input data for an instance.
- Explanation: As anomalies are special data instances with extraordinary properties, it would be very interesting to know why they are abnormal [1].

1.2 Problems in chemical anomaly detection

Chemical anomaly detection is the task of finding all anomalous molecules from a set $\mathcal{X}$. Therefore, it is a graph-level anomaly detection task, meaning that x_i is a graph representing a molecule. An additional challenge in chemical anomaly detection is that there is no precise definition of outliers in chemical space [2]. Therefore, a precise optimization problem cannot be formulated complicating the problem tremendously. An anomaly must be either defined or classification datasets have to be used to evaluate the tested outlier detection methods [10]. This work will use the latter approach and define its own datasets $\mathcal{X}$ with an associated labels $\mathcal{Y}$ based on knowing $\mathcal{P}$ in chapter 4 and it will show that classification on those datasets is easy whilst anomaly detection remains extremely difficult in chapter 5. In order to do this several anomaly detection experiments will be performed based on supervised experiments trained with $\widetilde{\mathcal{I}}$ whilst the classification techniques are trained on a subset of the labeled data. Further, there is the challenge of representing molecular graphs. In chapter 2 two different methods of representing molecular graphs will be discussed.

1.3 Contribution

The contribution of this thesis is the formalization of the problems in anomaly detection described by Pang G. et al. [1] and the subsequent construction of chemical datasets based on these challenges. Furthermore, it is shown that whilst classification, where anomalous and non-anomalous data points are available for training, is feasible for the constructed datasets, selected anomaly detection algorithms, which are only trained on inliers as described in section 1.1, still struggle. Finally, a simple anomaly detection algorithm is proposed that is able to solve the datasets showing that training on the inliers only, is not the problem.

2 | Preliminaries

This chapter describes two ways of how molecules can be represented in section 2.1. Furthermore, there is a description of graph neural networks in section 2.2. Section 2.3 describes dimensionality reduction, section 2.4 classical anomaly detection techniques, and section 2.5 describes frequent subgraph mining.

2.1 Molecular representation

In order to perform the graph anomaly detection task, molecules have to be encoded. This can be done in a multitude of different ways but representing them as graphs or via 2D fingerprints are popular options.

2.1.1 Graphs

A graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ consists of a set $\mathcal{V}$ of nodes (also called vertices) and a set of edges $\mathcal{E}$. Every node $\widetilde{v_i} \in \mathcal{V}$ has a label $v_i \in \mathbb{R}^n$ describing its properties. An edge connects two nodes and can be represented as a tuple of nodes $(\widetilde{v_i}, \widetilde{v_j}) \in \mathcal{E}$. The set of edges $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ can be stored in an adjacency matrix $\mathcal{A}$. $\mathcal{A}_{ij}$ is one if $(\widetilde{v_i}, \widetilde{v_j}) \in \mathcal{E}$ and else zero. It is possible for the same graph to have multiple adjacency matrices that are valid for different orderings of the vertices, because even if the nodes are ordered in a different way, the graph remains the same. Assuming that there are two graphs $\mathcal{G}_1$ and $\mathcal{G}_2$ with two arbitrarily ordered sets $\mathcal{V}_1$ and $\mathcal{V}_2$ as their vertex sets which contain nodes with the same labels, then there exists a permutation Π that maps $\mathcal{V}_1$ to $\mathcal{V}_2$. If and only if their adjacency matrices $\mathcal{A}_1$ and $\mathcal{A}_2$ satisfy Equation 2.1, Π is called an isomorphism and $\mathcal{G}_1$ and $\mathcal{G}_2$ are called isomorphic.

$$\Pi \mathcal{A}_1 \Pi^{-1} = \mathcal{A}_2 \tag{2.1}$$

Isomorphic means that the two graphs are identical or that there exists a bijective structure-preserving mapping between the two of them.

In molecules the direction of the bonds does not matter in most cases. Therefore, when representing a molecule as a graph with atoms as nodes and bonds as edges, it can be considered an undirected graph which means that if $\mathcal{E}$ contains $(\widetilde{v_i}, \widetilde{v_j})$ it also contains $(\widetilde{v_j}, \widetilde{v_i})$. Hence the adjacency matrix is symmetric. Intuitively, it makes sense to include physical properties like atom type and charge into the node labels v_i. Figure 2.1 shows how such an encoding could look like. Something else that has to be considered when encoding molecules as graphs is how to encode the different properties of the atoms. For example, the element can be defined by the number of protons in the core which is a numeric value or as a categorical value where the atomic number is one-hot encoded.

2.1.2 Fingerprints

It is also possible to represent a molecule as a vector. This type of representation is called molecular fingerprint. The features in the fingerprint can be thought of as physical properties in the

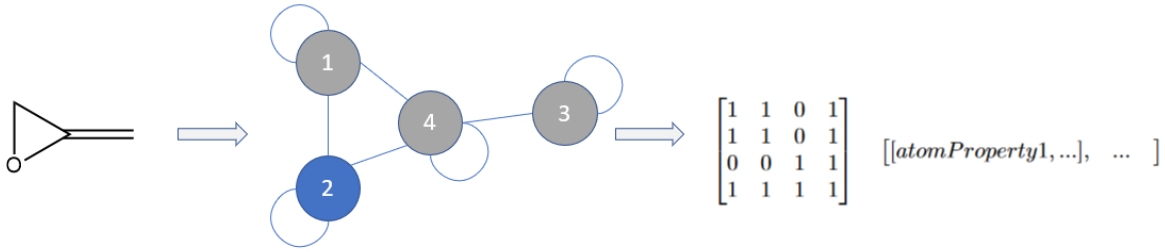

Figure 2.1: Generation of molecular graphs, self-loops are added

molecule. For example, solubility and molecular weight are features that are used often but there is also the possibility of encoding them such that the features include information about the occurrence/absence of a substructure in the molecule. Circular fingerprints are encoding molecule in that way: For every atom in a molecule and for every radius between 0 and r, all possible atomic neighborhoods are calculated. Afterwards, each unique path is hashed, and the corresponding bit is activated. Figure 2.2 visualizes this process [11].

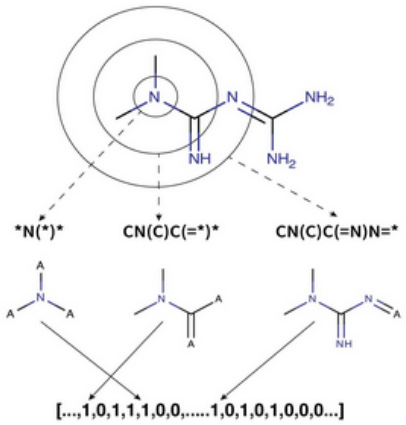

Figure 2.2: Generation of circular fingerprint [11]

2.2 Graph neural networks

A graph neutral network (GNN) is an artificial neural network that deals with graph data. The idea behind it is that each vertex $\widetilde{v}_i$ receives information from its immediate neighborhood $\mathcal{N}(\widetilde{v}_i) = \{\widetilde{v}_j \in \mathcal{V} \mid (\widetilde{v}_i, \widetilde{v}_j) \in \mathcal{E}\}$ and updates its label v_i accordingly. Often self-connections are added such that a node is able to keep information from the previous update step. The procedure of passing information is visualized in Figure 2.3. After one iteration of passing information to its neighbors, the labels of the nodes (visualized by color) are changed according to the information of the neighboring nodes. This means that in the next iteration nodes also receive information from nodes that are not in their immediate neighborhood because now the information that their neighbors contain was augmented by the information of their neighbors. When doing these computations, one must take into account that nodes can be labeled arbitrarily therefore information passing must consider this. The entire network, including the loss, has to be permutation invariant, which means that the output does not change with the ordering of the input, to achieve this the final layer is often a global pooling layer but the individual layers in between only have to be permutation equivariant, which means that if one provides two isomorphic graphs as inputs, then the output

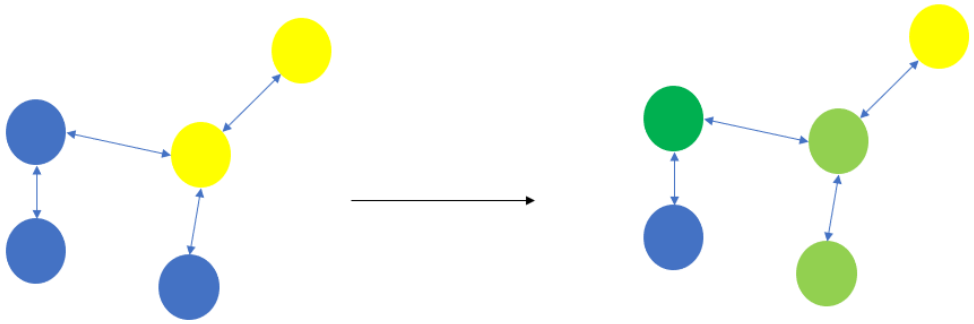

Figure 2.3: Message passing in GNNs, the colors represent the information

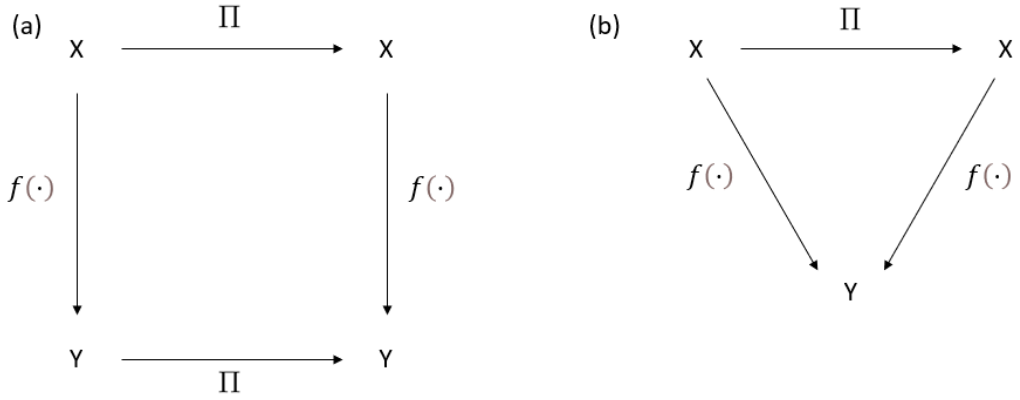

Figure 2.4: (a) permutation equivariance; (b) permutation invariance

is equal up to permutation and that permutation is the isomorphism. Figure 2.4 visualizes what it means for $f(\cdot)$ to be permutation equivariant (a) and permutation invariant(b). The most commonly used computational module of GNNs are:

- Propagation Module
- Sampling Module
- Pooling Module

The propagation module is responsible for the passing of information the nodes to each other. It has to be permutation equivariant [12]. An example for such an operation can be seen in Equation 2.2. It is called graph convolutional operator [13].

$$H^l = \sigma(\tilde{D}^{-\frac{1}{2}}\mathcal{A}\tilde{D}^{-\frac{1}{2}}H^{l-1}\Theta^{l-1}) \tag{2.2}$$

H^l is the output of layer l in the network. H^0 is a tensor containing the information of the node labels of the currently processed graph. $\mathcal{A}$ is the adjacency matrix of the graph that is currently processed with self-loops. $\tilde{D}$ is diagonal and D_{ii} says how many edges node $\widetilde{v}_i$ has. It is called degree matrix and helps with normalization. Θ^{l-1} is the matrix containing the learnable weights for layer $l-1$. $\sigma(\cdot)$ is an activation function [12]. When considering a permuted input, we get:

$$\begin{aligned}\widetilde{H^l} =&\sigma(\Pi\tilde{D}^{-\frac{1}{2}}\Pi^{-1}\Pi\mathcal{A}\Pi^{-1}\Pi\tilde{D}^{-\frac{1}{2}}\Pi^{-1}\Pi H^{l-1}\Theta^{l-1})\\ =&\Pi\sigma(\tilde{D}^{-\frac{1}{2}}\mathcal{A}\tilde{D}^{-\frac{1}{2}}H^{l-1}\Theta^{l-1})\\ =&\Pi H^l\end{aligned} \tag{2.3}$$

Permuting the input means permuting the adjacency matrix, the degree matrix and the outputs of the previous layer/the initial input. Because the activation function is applied element-wise, one can factor out the permutation matrix. Therefore, if the ordering of the input graph is permuted by Π, the output is also permuted by Π, which means that this operation is permutation equivariant.

Sampling modules are required when graphs are large, and the networks are deep. Because the neighbors that have to be taken implicitly into account grows exponentially with the depth of the network. Because molecules are rather small graphs such modules are not required for graph level anomaly detection for molecules.

Pooling modules aggregate the information passed through the network in order to obtain more general features. There are direct pooling operations and hierarchical pooling operations. An example for a simple pooling operation is max pooling where one gets a graph representation by taking the maximum over the channel dimension. Notably, this operation is permutation invariant and is therefore often used as a readout operation [12].

2.2.1 Hierarchical Graph Representation Learning with Differentiable Pooling

DiffPool is a powerful pooling module. Ying, R. et al. [12] introduced this hierarchical pooling operation DiffPool for graph neural networks. DiffPool learns how to map nodes of an input graph onto a set of clusters, which in turn form a coarsened input for the next layer. With other words, the goal is to find a GNN module that learns how to take a graph, represented by a node embedding and adjacency matrix $A \in \mathbb{R}^{n\times n}$, and transform it into a smaller graph with a weighted adjacency matrix $A' \in \mathbb{R}^{m\times m}$ and node embeddings $Z' \in \mathbb{R}^{m\times d}$ where $m < n$, which can be subsequently used by other GNN modules.

This is done by using an assignment matrix $S^{(l)} \in \mathbb{R}^{n_l\times n_{l+1}}$. Each row of $S^{(l)}$ corresponds to a node in the current graph and each column to a node in the new smaller graph. The number of nodes n_i in layer i are hyperparameters of the new resulting network architecture. The assignment matrix is calculated by the row-wise softmax of a separate GNN module that contains learnable parameters as can be seen in Equation 2.4.

$$S^{(l)} = \text{softmax}\Big(\text{GNN}_{\text{pool},(l)}\big(A^{(l)}; X^{(l)}\big)\Big) \tag{2.4}$$

$A^{(l)}$ is the adjacency matrix in layer l and $X^{(l)}$ is the node embedding in layer l. They are the input into the DiffPool module. Therefore, the entry $S^{(l)}_{ij}$ can be interpreted as the probability of node i belonging to node j in the coarsened graph. Additionally, normal information propagation takes place with another GNN module:

$$Z^{(l)} = \text{GNN}_{\text{embed},(l)}\big(A^{(l)}; X^{(l)}\big) \tag{2.5}$$

Finally, the output of the DiffPool layer is calculated by

$$X^{(l+1)} = \big(S^{(l)}\big)^T Z^{(l)} \tag{2.6}$$

$$A^{(l+1)} = \big(S^{(l)}\big)^T A^{(l)} S^{(l)} \tag{2.7}$$

Because the gradient for updating the pooling graph neural network might not be sufficient to escape early local minima, two additional loss terms are added: the link prediction loss term, which encodes the notion that neighboring nodes should remain close together with $L_{LP} = ||A^{(l)} - \left(S^{(l)}\right)^T S^{(l)}||_F$ and a row wise entropy term on the assignment matrix such that definitive assignments are preferred over assignments with high entropy. With this approach the accuracy of graph classification benchmarks is improved upon by 5% to 10% [14].

2.3 Dimensionality reduction

Complex data carries a lot of useful information but obtaining useful information from such data becomes increasingly difficult with growing dimensionality. This effect is known as "curse of dimensionality" and stems from the fact that with increasing dimensions the data becomes more and more sparse making it difficult to learn the relevant features for the task at hand. The task of dimensionality reduction is to decrease the number of dimensions a data instance has whilst keeping the main features. This means that a n-dimensional vector should be mapped to a m-dimensional vector where $n \gg m$. There is feature selection and feature extraction. In feature selection a subset of the original features is kept, and the rest is discarded, while in feature extraction the data is transformed such that the new dimensionality is much lower than the old one [15].

2.3.1 PCA

Principal component analysis (PCA) is one of the most prominent feature extraction techniques. PCA finds an orthonormal basis where the average distance from every point to its projection is minimized. At the same time, this procedure maximizes the variance of the projected data. [15] There are N data points with dimensionality n with $x_i \in \mathbb{R}^n$ being data instance i. X is a matrix containing the entries of x_i. $\hat{x_i} \in \mathbb{R}^m$ is the projected data instance i by the orthonormal basis $W = (w_1|\dots|w_n)$ associated with $\mathrm{Cov}[X]$ which exists because covariance matrices are positive semidefinite. Assuming zero-centered data the covariance matrix of the projected data can be described as in Equation 2.8

$$\begin{aligned} \mathrm{Cov}[W^\top X] =& \mathrm{E}[W^\top X (W^\top X)^\top] \\ =& W^\top \mathrm{E}[X X^\top] W \\ =& W^\top \mathrm{Cov}[X] W \end{aligned} \tag{2.8}$$

The variance, which is equal to the diagonal entries of the covariance matrix, of the projected data along w_i is maximized and it is taken into account that $\|w_i\|_2^2 = 1$. This can be done by the use of Lagrange multipliers as shown in Equation 2.9.

$$\begin{aligned} \frac{\partial \mathrm{Var}[W^\top X] + \lambda_i (1 - w_i^\top w_i)}{\partial w_i} =& \frac{\partial w_i^\top \mathrm{Cov}[X] w_i + \lambda_i (1 - w_i^\top w_i)}{\partial w_i} \\ =& 2\mathrm{Cov}[X] w_i - 2\lambda_i w_i \\ =& 0 \end{aligned} \tag{2.9}$$

Equation 2.9 shows that the orthonormal basis has to be composed of the eigenvectors of the covariance matrix of the original data. The eigenvector with the largest associated variance and hereby eigenvalue is called first principal component. If one wants to use PCA a dimensionality reduction tool, the dimensions which are associated with the lowest variance are emitted when calculating the transformed data. This is typically done in such a way that at least certain

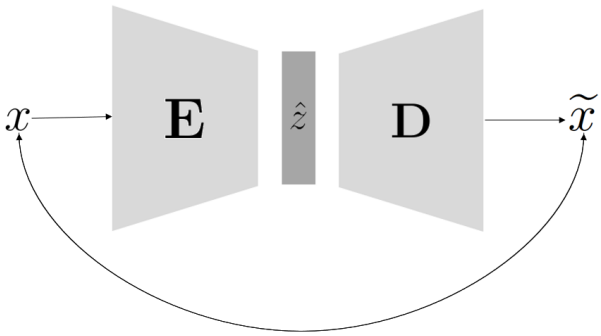

Figure 2.5: Schematic of an autoencoder

percentage of total variance is kept. PCA removes linear dependencies between the axis of the original data but the result changes depending on the scaling of the different variables. However, this can be solved by standardizing the variables which means transforming the data to data of a standard normal distribution. In theory there is no problem if the number of features is bigger than the number of data point. It implies that some eigenvalues are zero in the covariance matrix because the rank of the data X and the covariance matrix have to be equal, and the rank is bound by the smaller dimension of a matrix. But it has been demonstrated that in the case where there are few samples compared to the features the principal components do not correspond well to the initial dimensions anymore [16].

2.3.2 Autoencoder

An autoencoder is a neural network that tries to reconstruct its input x by comparing it to its output $\widetilde{x}$. Figure 2.5 displays a schematic of an autoencoder. There is an encoder **E** and a decoder **D**. In order to learn the important features, the dimensionality in the center of the autoencoder is usually smaller than the input dimensions. This type of autoencoder is called undercomplete. The shortest vector representation directly after the encoder **E** is a compressed version of the input and it lies in the so called latent or embedding space and is often called $\hat{z}$. After training the autoencoder, the encoder can be used for dimensionality reduction. The encoder retains most important information of the data whilst compressing it. This procedure is also a feature extraction technique. In contrast to PCA not only linear combinations of the features are extracted but also non-linear coherences are learned because of the non-linear nature of the activation functions [17]. The reconstruction error of autoencoders can be used as an outlier score [18].

2.4 Classical anomaly detection techniques

The classical anomaly detection techniques that are described in this section are applied on vector data $x \in \mathbb{R}^n$. Algorithms like DBScan or Local Outlier Factor (LoF) assume that anomalous data instances occur more rarely and therefore the region in which they lie is sparse. Isolation Forest (iForest) follows a similar notion by assuming that outliers can be isolated easily by random splits along the axis of the coordinate system. One class support vector machines (OneClassSVM) on the other hand builds a hyper sphere that contains all inliers (which are available for training) within error margins. This assumes that there is a center of the normal data.

2.4.1 Density-based anomaly detection

Density-based anomaly detection techniques are used to identify anomalous data points in a dataset based on their deviation from the overall data density. The fundamental idea is that normal data points are expected to be densely packed in the feature space, while anomalies are expected to be located in regions of lower density. DBScan and Local Outlier Factor are examples for density-based anomaly detection techniques. They do not rely on predefined thresholds or assumptions about the data distribution, making them more robust to different types of data and anomalies. They are also capable of detecting anomalies in both global and local contexts, allowing them to capture different types of anomalies such as global outliers or local clusters of anomalies [19].

DBScan

In DBScan there are two important hyper parameters: ε and $minPts$. A data point p is called a core point if at least $minPts$ are within a distance of ε. A point q is called directly reachable if it is within distance ε of a core point. A point p_n is called reachable from p_1 if there is a path $p_1, \dots p_n$ on which p_{i+1} is directly reachable from p_i. If p is a core point it forms a cluster with all points that are reachable from it. All points that are not reachable from any core point are anomalies. These anomalies are either too far away from other points or there are only few points surrounding it, but this group is not sufficiently large to produce a core point. These could be considered group anomalies. Typically, the Euclidean distance is used to calculate the distance between instances, but it is also possible to use a different metric. DBScan does not consider variation in density [20].

Local Outlier Factor

Local Outlier Factor is an algorithm that compares the density of a data point with its surrounding. If it is lower, then the anomaly score that results from the algorithm is higher. For every data point its k-nearest neighbors are searched and the distance to the k^{th}-nearest neighbor (k^{th}- dist$(\cdot)$) is calculated. Afterwards, the reachability distances (d_{reach}) from every neighbor B_i to every point A is calculated as can be seen in Equation 2.10.

$$d_{reach}(A; B_i) = \max\left\{k^{\text{th}}\text{- dist}(B_i); d(A, B_i)\right\} \tag{2.10}$$

The inverse of the average reachability distances of a point A is called the local reachability density. Finally, to calculate the anomaly score the local reachability density of a point is compared to the average of its neighborhood by dividing the average of the neighborhood by the local reachability density of the point in question [21].

2.4.2 Isolation Forest

Isolation Forests decides whether a data instance is an anomaly or not based on an ensemble decision of isolation trees. To build an isolation tree (iTree) a random subset of the training data is chosen. The space is split recursively by selecting a feature and a split value at random and a binary tree is built. This procedure is concerted until every node/subspace contains only one instance from the subset. Data instances that can easily be split up from the rest are more anomalous (far up in the isolation tree). Figure 2.6 shows an example of an isolation tree. [22].

2.4.3 OneClassSVM

In a one class support vector machine, the goal is to find a hyper sphere that encompasses all training examples. In order to account for training points that lie outside of this sphere by chance,

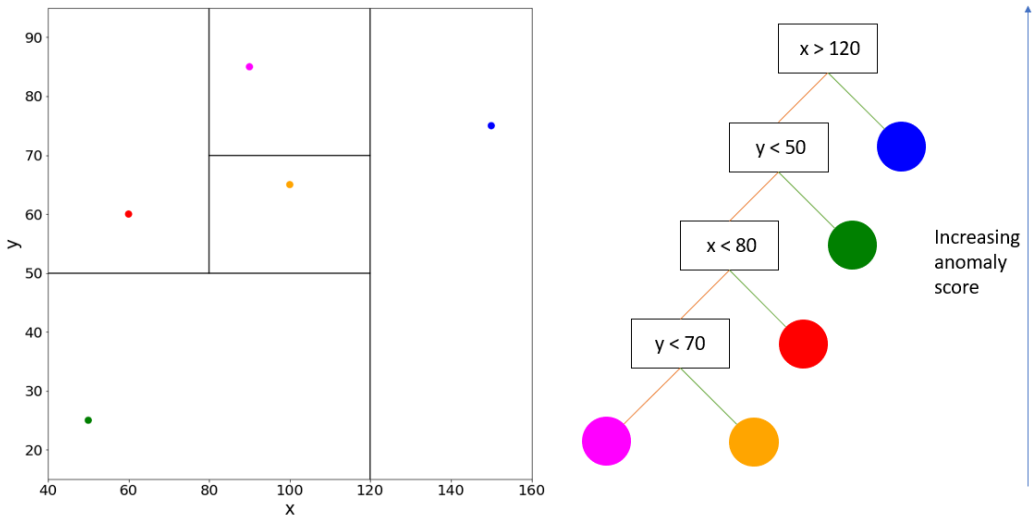

Figure 2.6: Example of an isolation tree

slack variables are introduced that allow for a margin of error for every data point. With these considerations in mind the problem can be formulated as can be seen in Equation 2.11

$$F(R,a) = R^2 + C\sum \xi_i \qquad \text{constraint by} \qquad ||x_i - a|| \leq R^2 + \xi_i \tag{2.11}$$

C is a hyper parameter that controls the trade off between the volume of the sphere and the error, R is the radius of the sphere, ξ_i is the error made by the i^{th} training instance x_i, and a is the center of the sphere. The constraints can be incorporated into the problem through Lagrange multipliers and the problem can be reformulated such that it is kernalizable. This means that the problem is only dependent on inner products of the training data and allows the use of a transformed domain, which fits the problem better, for the calculation of the inner product. With other words, instead of $K(x_i, x_j) = \langle x_i, x_j \rangle$, $\widetilde{K}(x_i, x_j) = \langle \psi(x_i), \psi(x_j) \rangle$ with $\psi : \chi \to \rho$ can be used and ψ does not have to be known because knowing how to calculate $\widetilde{K}(\cdot, \cdot)$ is sufficient. This allows for non-spherical decision boundaries in χ. A possible kernel is the Gaussian Kernel which is defined $\widetilde{K}(x_i, x_j) = \exp\left(\frac{-||x_i - x_j||}{s^2}\right)$, where s is a hyper parameter that determines how curlicued the decision boundary is. Small s lead to more squiggly decision boundaries making the one class support vector machine more powerful but also more prone to overfitting [23].

2.5 Frequent subgraph mining

Frequent subgraph mining (FSM) is the task of finding all subgraphs in a graph or a list of graphs that appear more often than a threshold (support threshold). These graphs are called frequent. If only a single graph is searched, the problem is called single graph-based FSM and if a list of graphs is searched it is called a graph transaction-based FSM. In graph transaction-based FSM it is either possible to count all occurrences of a subgraph in all graphs or to count the number of graphs that contain that subgraph. The latter approach is called transaction-based counting and the support of a subgraph g is defined by the relative number of graphs that contain that subgraph g. Transaction-based counting allows the use of the downward closure property. The downward closure property states that if a graph is frequent, then all its subgraphs are necessarily frequent [24]. Figure 2.7 displays cubane, whose molecular graph is an example why the downward closure property does not hold, when counting the number of occurrences of a subgraph g for determining its support. With a support threshold of 10 the subgraph "C-C" is frequent with a

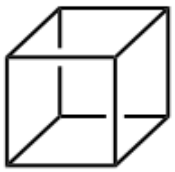

Figure 2.7: Cubane as a counterexample for the downward closure property with occurrence-based counting

support of 12, which is equal to the number of edges of the cube. "C" on the other hand is not frequent with a support of 8, which is equal to the number of corners. Canonical labeling of graphs is the act of converting a graph G as described in subsection 2.1.1 to a sequence such that all graphs that are isomorphic to G obtain the same new representation. An example for canonical labeling of graphs is the minimal DFS code.

2.5.1 Minimal DFS code

Minimal DFS code is a canonical labeling based on depth-first search. This is done by first constructing DFS trees and finding their DFS codes. Figure 2.8 shows three possible depth-first search trees for isomorphic graphs and Table 2.1 shows the resulting DFS codes.

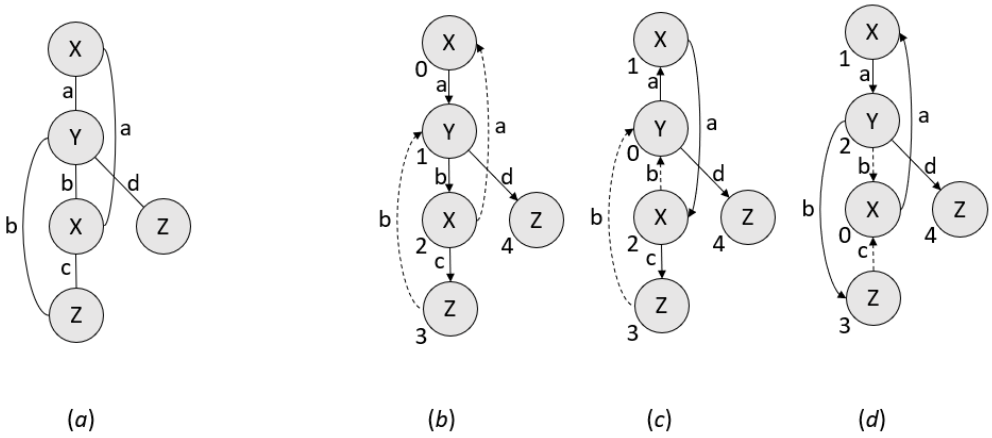

Figure 2.8: DFS trees ((b)-(d)) for graph (a). The numbers show the order with which the nodes are discovered. The dashed lines show the edges to nodes previously visited.

Table 2.1: DFS codes for DFS trees from Figure 2.8

#	Figure 2.8 (b)	Figure 2.8 (c)	Figure 2.8 (c)
0	(0,1,X,a,Y)	(0,1,Y,a,X)	(0,1,X,a,X)
1	(1,2,Y,b,X)	(1,2,X,a,X)	(1,2,X,a,Y)
2	(2,0,X,a,X)	(2,0,X,b,Y)	(2,0,Y,b,X)
3	(2,3,X,c,Z)	(2,3,X,c,Z)	(2,3,Y,b,Z)
4	(3,1,Z,b,Y)	(3,0,Z,b,Y)	(3,0,Z,c,X)
5	(1,4,Y,d,Z)	(0,4,Y,d,Z)	(2,4,Y,d,Z)

With the following three rules, the edges $e_1 = (i_1, j_1)$ and $e_2 = (i_2, j_2)$ can be linearly ordered according to the discovery time of their nodes i and j, where i was always discovered first e.g. $i < j$. This linear ordering is called $\prec_T$

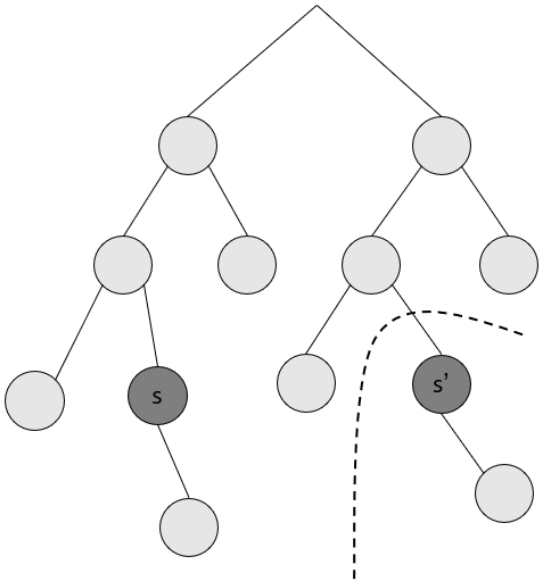

Figure 2.9: DFS Code tree. The node s' is isomorphic to s but can be pruned because its DFS code is not minimal.

$$
\begin{aligned}
i_1 = i_2 \text{ and } j_1 < j_2 &\quad \Rightarrow e_1 \prec_T e_2 \\
i_1 < j_1 \text{ and } j_1 = i_2 &\quad \Rightarrow e_1 \prec_T e_2 \\
e_1 \prec_T e_2 \text{ and } e_2 \prec_T e_3 &\quad \Rightarrow e_1 \prec_T e_3
\end{aligned}
$$

The DFS code is an edge sequence based on a DFS tree for a graph G that is constructed based on $\prec_T$ such that $e_i \prec_T e_{i+1}$ for all edges in the graph. For example, such an edge can be represented by a tuple of node discovery times, node types, and edge type. This edge representation is also used in Table 2.1. The straight path from the first to the last discovered node (rightmost vertex) is called rightmost path and consists only of forwards edges, which means that all edges $e_i = (v_n, v_m)$ on this path have $n < m$. Furthermore, DFS codes can also be linearly ordered with $\prec_e$ if the label set has a linear ordering (the node types and edge types can be ordered) [25]. DFS Lexicographic Order is a linear order and is defined as follows: If and only if there is a t between zero and the smaller number of edges ($\min(n, m)$) of the two graphs for the DFS codes $\alpha = (a_0, \ldots, a_m)$ and $\beta = (b_0, \ldots, b_n)$ such that all previous edges are the same and $a_t \prec_e b_t$ or all edges up until m are the same and $m \leq n$, then $\alpha \leq \beta$. The DFS code with the minimal DFS lexicographic order of a DFS tree of a graph is called the Minimum DFS Code of G and is a canonical label of G.

Given DFS codes $\alpha = (a_0, \ldots, a_m)$ and $\beta = (a_0, \ldots, a_m, b)$, one considers β to be α's child and α in turn is β's parent. Further, b must be an edge which grows from the vertices on the rightmost path, because otherwise rules for constructing a valid DFS code are violated. Additionally, if α is a minimum DFS code and $a_m \prec_e b$, then β is also a minimum DFS code. From this relation, one can create a DFS Code tree, which essentially spans the search space, by extending the graphs one edge at a time. Figure 2.9 shows how the DFS Code tree is grown. If the DFS code of s' is not minimal and the DFS Code tree is searched with a depth first approach, it can be pruned because an isomorphic graph s with minimal DFS code has already been found previously.

2.5.2 gSpan: Graph-Based Substructure Pattern Mining

gSpan is a transaction-based counting FSM algorithm that utilizes a depth-first search approach. For efficient enumeration of the candidate frequent subgraphs, the idea of a DFS tree is used. Algorithm 1 shows the gSpan and algorithm 2 shows the procedure of subgraph mining. For each edge from the set of frequent 1-edge subgraphs in lexicographic order the subgraph mining procedure is called with the arguments of subgraph that is currently checked (the current edge), all graphs that contain that subgraph, and all frequent subgraphs (all edges). In the subgraph mining procedure, it is first checked whether the current subgraph has a minimal DFS code. If it does not have a minimal DFS code, then the subgraph itself (because of the lexicographic order) and all its descendants (because of the downward closure property) have already been checked and the DFS code tree can be pruned here. Afterwards, the subgraph is added to the set of frequent subgraphs and all its children are enumerated with one edge growth. For each child the support is checked and if the support is above the threshold the subgraph mining procedure is called recursively with the child being the new frequent subgraph and all graphs containing that subgraph being the new set of subgraphs. Afterwards, the frequent 1-edge that was used to start the recursion process from all graphs, is removed from the dataset because all frequent subgraphs containing this edge have already been mined. With this approach massive speedups are achieved on synthetic and real-world data sets compared to the breadth-first FSM algorithm FSG [26, 27].

Algorithm 1 gSpan($\mathbb{D}$)

```
sort the labels in 𝔻 by their frequency
remove infrequent vertices and edges
relabel the remaining vertices and edges
𝕊¹ ← all frequent 1-edge graphs in 𝔻
sort 𝕊¹ in DFS lexicographic order
𝕊 ← 𝕊¹
for e ∈ 𝕊¹ do
    initialize s with e with s.𝔻 = {g|∀g ∈ 𝔻, e ∈ E(g)}
    Subgraph_Mining(𝔻, 𝕊, s)
    𝔻 ← 𝔻 − e
    if |𝔻| < minSupport then
        break
    end if
end for
```

Algorithm 2 Subgraph_Mining($\mathbb{D}$, $\mathbb{S}$, s)

```
if s ≠ min(s) then
    return
end if
𝕊 ← 𝕊 ∪ {s}
enumerate s in each graph in 𝔻 and count its children
for c, c is s' child do
    if minSupport ≤ support(c) then
        s ← c
        Subgraph_Mining(𝔻_s, 𝕊, s)
    end if
```

3 | Related Work

This chapter will describe the approach of Casalegno M. et al. [2] and explain why it tackles the problems of unlikeness well whilst the ideas of unknowness and irregularity are not incorporated into their algorithm. Furthermore, a classification of anomaly detection techniques by Pang G. et al. [1] is presented to pick representative algorithms for each class for further experimentation. Lastly, a graph neural network-based graph level anomaly detection method will be presented.

3.1 Definition and Detection of Outliers in Chemical Space

The basic idea behind the algorithm of Casalegno M. et al. [2] is to transform a given fragment-based representation of molecules into a cluster-based one. In order to find out how much each cluster contributes to each of the N molecules, an iterative procedure is used. If the fragment j occurs more often than once, a set $C(j)$ with all molecules containing fragment j is created. In addition to these sets, it is also possible that each molecule belongs to an outlier class, which in turn would make it an anomaly. These are the classes onto which the algorithm tries to map the fragment-based representation. This notion of each anomalous molecule belonging to its own class reflects the problem of unlikeness very well. $m(i)$ is the number of clusters molecule i belongs to after removal of features that only occur once. With this information, a membership matrix $P \in [0,1]^{N \times M+1}$ can be created, where the entry P_{ij} can be interpreted as a probability of molecule i belonging to cluster j. The cluster $j = 0$ is the outlier cluster. If and only if $P_{i0} = 1$ molecule i is considered to be an outlier. Additionally, an affinity matrix A, whose entry A_{ij} describes the affinity of molecule i to the other molecules in the cluster, and a weight matrix W, which contains information about how important a molecule is for the representation of a cluster, are required for the iterative procedure shown in algorithm 3. Additionally, there is the similarity matrix S, which contains information about how similar molecules are by the Tanimoto similarity. The matrices with the superscripts are updated. The approach of Casalegno M. et al. identifies compounds without specific structural features, rather than compounds with rare features. This algorithm is applied to several databases and up to 13.5% of compounds are identified as anomalies [2].

However, taking the problems in anomaly detection from Pang G. et al. [1] into consideration there are two problems with this approach: Firstly, the correct identification of group anomalies is unlikely. If a feature occurs more than once, a set to which molecules can be classified to, is created. All group anomalies (of a certain group) are assigned to this class instead of the outlier class, hereby leading to incorrect identification. For example, all inliers are not metals, but there are two metals and there is the feature "is metallic". The metals would be put into the "is metallic" class because they have a large Tanimoto similarity, instead of the outlier class.

Secondly, the problem of unknowness is not incorporated into this approach. If additional anomalies become available for training, the result of algorithm changes because new classes might become available. For example, similar as above, what happens when there is only one metal and then an additional metal becomes available for training. Normally, unknowingly training on anomalies in an unsupervised setting does not pose a problem because of rarity, but the proposed algorithm identifies a large percentage of molecules as anomalous, making the unwanted new

Algorithm 3 FindOutlier($\mathbb{D}$)

Calculate the similarity matrix S of all molecules
Initialize P^0 with a uniform distribution of across all features that a molecule contains and the outlier class
Initialize W^0 with 1 where it belongs to a created class (from fragments and the outlier class) and 0 where it does not
Compute all $C(j)$ and $m(i)$
Select convergence criterions
$t = 1$, $j = \{1, \ldots, M\}$, $i = \{0, \ldots, N-1\}$
while convergence criterions not satisfied **do**
$A_{ij}^t = \frac{\sum_{k \in C(j)} S_{ik} \dot{W}_{kj}^{t-1}}{|C(j)|-1}$
$A_{i0}^t = P_{i0}^{t-1}$
$P_{ij}^t = \frac{A_{ij}^t}{A_{i0}^t + \sum_{k=1}^{M} A_{ik}^t}$
$P_{i0}^t = \frac{A_{i0}^t}{A_{i0}^t + \sum_{k=1}^{M} A_{ik}^t}$
$W_{ij}^t = (m(i)+1) \cdot P_{ij}^t$
$W_{i0}^t = (m(i)+1) \cdot P_{i0}^t$
t+=1

class creation relatively likely. In order to better tackle the problem of unknowness, the anomaly detection algorithms in this thesis only train on inliers.

3.2 Categorization of Deep Anomaly Detection

There are three conceptual approaches for deep anomaly detection:

- Learning for feature extraction
- Learning feature representations of normality
- End-to-end anomaly score learning

In deep learning for feature extraction the objective is to extract a lower dimensional representation for the data. This representation is subsequently used for an independent anomaly scoring algorithm. The major assumption that one makes when using this paradigm is that the learned features preserve the information that is able to differentiate between anomalous and normal instances.

In contrast, when learning feature representations of normality, the anomaly scoring, and the feature extraction are not decoupled. In this category the anomaly scoring is a direct result of the measure-driven loss function. Furthermore, one can differentiate between generic normality feature learning and anomaly measure-dependent feature learning. In the former a general representation of the data is learned whilst in the latter the representation that is learned is optimized for an already existing anomaly measure.

Finally, there is end-to-end anomaly scoring in which a neural network learns anomaly scores directly. The main difference between learning feature representation of normality and end-to-end anomaly scoring is that the latter focuses on finding new loss functions to score anomalies directly and the former tries to combine neural networks with a pre-existing anomaly measure [1].

3.3 Deep Graph-level Anomaly Detection by Glocal Knowledge Distillation

The paper deep graph-level anomaly detection by GLOCAL Knowledge Distillation by Ma R. et al. [28] introduces an approach that uses local and global graph information in order to predict anomalous graphs from 16 real-world datasets. This is done by using the idea of random knowledge distillation. In random knowledge distillation, there are two networks both of which are initialized randomly. One of them is called random target network and the other one is called predictor network. During training only the predictor network is trained. It tries to mimic the output of the random target network by minimizing the MSE between the outputs of the two networks when both networks are fed the same input. Therefore, labels are not required in this type of architecture. Additionally, it is assumed that novel inputs lead to a larger prediction error than known inputs [29]. Ma R. et al. use this idea in combination with graph neural networks. They use two GNNs with graph convolution layers, one as a random target and one for prediction and then train the prediction network on the inliers only. As discussed in chapter 1 this is done in order to ensure that the network does not have information about how anomalies look like. Further, local and global information is incorporated into the loss function by comparing the random target network once after the graph convolution layers and once after a final pooling layer in order to get more global information. Figure 3.1 shows the architecture of the network. Notably, the loss function ensures that the entire network is permutation invariant if the layers connecting start to end are permutation equivariant, because the permutation equivariance ensures that the output of both networks is permuted in the same way and therefore, the same nodes are always compared regardless of the permutation of the input. Their approach is able to achieve better performance than several state-of-the-art techniques but the AUC of the 16 real-world datasets still ranges from 0.514±0.039 to 0.992±0.004. This means that the success of their approach is still heavily dependent on the dataset which is used [28].

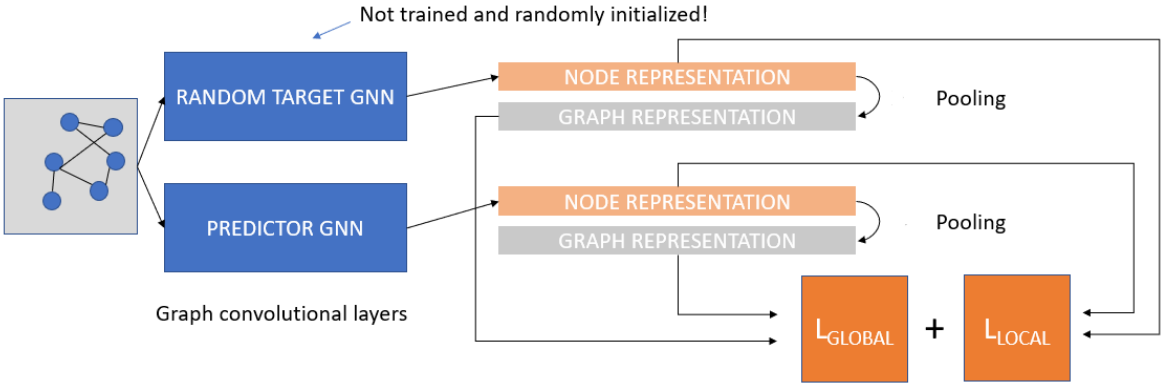

Figure 3.1: Architecture of GLOCAL anomaly detection

3.4 Raising the Bar in Graph-level Anomaly Detection

Qiu C. et al. [10] develop OCGTL, a novel end-to-end method for graph-level anomaly detection that tries to overcome the challenges of hypersphere collapse and performance flip by jointly training multiple GNNs. Performance flip is a problem that can occur if classification datasets are used for an anomaly detection problem. It describes the discrepancy in performance between two experimental setups, where the only difference is the class that is used as an inlier. Hypersphere collapse is a problem in one class classification that can occur if the goal of the training is

to find an embedding ϕ that projects all inliers to a similar region. If the hypersphere collapses then $\phi(x_i; W) = c$, with x_i being the training example i, W being the weights of the algorithm, and c being constant - with other words, the output of the embedding is irrespective of the input and therefore, the anomalies are also projected to that region of space making outlier detection impossible [30]. Qiu C. et al. propose a network architecture in which it is theoretically impossible for the hypersphere to collapse. In this approach, contrary to other methods, the center of the hypersphere is a learnable parameter, and the trivial solution is prevented by a regularization term. There is one reference network and $K + 1$ GNNs in total. The regularization term ensures that the non-reference networks result in dissimilar representations to each other whilst outputting a similar representation to the reference network. As a similarity measure the cosine similarity is chosen. With this approach, they are able to achieve AUCs of 9 real-world datasets ranging from 0.607±0.024 to 0.975±0.020. They are also able to show that their architecture does not have a problem with performance flip in practice.

3.5 Benchmark datasets

Table 3.1 shows the benchmark datasets that are used by Qiu C. et al. [10] (OCGTL) and Ma R. et al. [28] (GLOCAL) and their respective performances. The dataset REDDIT-B contains only two of five available classes.

Table 3.1: Commonly used graph-level anomaly detection datasets

Dataset	Domain	$\text{AUC}_{\text{GLOCAL}}$	$\text{AUC}_{\text{OCGTL}}$
DD [31]	Protein structure	**0.805**±0.017	0.699±0.026
PROTEINS [32]	Protein structure	**0.785**±0.034	0.607±0.024
ENZYMES [33]	Protein structure	0.636±0.061	**0.655**±0.038
AIDS [34]	Molecular	**0.992**±0.004	0.975±0.020
NCI1 [35]	Molecular	**0.683**±0.015	0.637±0.012
IMDB-B [36]	Collaboration network	0.514±0.039	**0.651**±0.018
REDDIT-B [36]	Discussion thread	**0.782**±0.016	0.774±0.019

4 | Methods

This chapter will outline how the test sets for anomaly detection are constructed. Furthermore, the three experimental approaches are described. Firstly, two classification experiments are specified in order to show that the datasets are feasible if anomalies are available for training. Secondly, three anomaly detection techniques, representatives of three different paradigms outlined in section 3.2, are selected and their hyperparameters listed, to test the performance of current anomaly detection techniques. Additionally, the GNN-based approach is tried to be approved upon with DiffPool. Finally, a novel anomaly detection approach is described that is able to solve the anomaly detection problem of the synthetic datasets.

4.1 Datasets

In order to test different approaches for graph anomaly detection in chemistry, test datasets are required. On the one hand, real-world classification datasets of molecules can be used for anomaly detection by simply emitting one class from the training set as discussed in chapter 1, on the other hand it is also possible to create your own classification datasets by deciding upon the property $\mathcal{P}$ and then selecting random molecules from a set that does not have this property as anomalies and again only training on the inliers. With the latter approach, one has more control over the unlikeness.

4.1.1 AIDS

The AIDS dataset consists of 2000 graphs representing active and inactive compounds against HIV. The atoms are nodes with information about element, charge, and position in a two-dimensional space. 38 different elements are present in the dataset. There are 1600 inactive and 400 active compounds. This dataset is relatively easy to classify as literature reports classification accuracies of 97.3% [34] and even anomaly detection algorithms score very well with an AUC of 0.992 ± 0.004 [28]. This dataset is used for GNN algorithms [37].

4.1.2 Drugbank

Drugbank is a comprehensive online database that provides information on drugs and drug targets. It contains detailed information about pharmaceuticals, including their chemical structure, pharmacology, pharmacokinetics, indications, contraindications, and side effects. Their database is collection of around 10000 small molecules. No labels are extracted from the Drugbank but rather labels are automatically annotated as described in subsection 4.1.3 [38].

4.1.3 Synthetic datasets

In order to create labels for the small molecules from the drugbank, the defining property $\mathcal{P}$ of the inliers is that at least one out of a set $\mathcal{S}$ of substructure is contained in the molecule. The anomalies

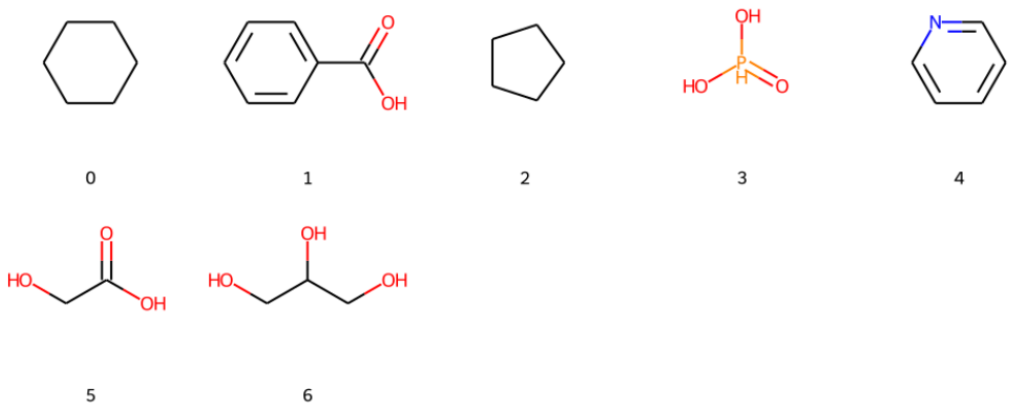

Figure 4.1: Substructures used to determine the synthetic datasets

are drawn randomly from the remaining molecules, which do not contain any substructure present in the set $\mathcal{S}$, with their number equal to a contamination parameter times the number of inliers. The contamination is a percentage of the number of inliers. It defines how many anomalies are added to each dataset.

Seven substructures are selected such that the number of molecules containing that substructure in the drugbank is higher than 400 but lower than 2500. Figure 4.1 displays these substructures. For constructing sets $\mathcal{S}$, two substructures are selected, and these sets are subsequently used to create datasets. The size of these set is chosen to be two because in most cases a class of chemicals is formed by several smaller subgroups, defined by a scaffold/substructure, and in order to reflect this and for the problem to be as simple as possible the number is chosen to be two. If the resulting datasets are too easy, then it is possible to select more substructures to create more difficult and larger datasets. To create a test dataset the following procedure is used:

1. Select a value of "contamination" and a set $\mathcal{S}$ with two of the substructures from Figure 4.1
2. Put all molecules from the drugbank that contain at least one of the two selected substructures into the dataset and label them as inliers
3. Multiply the number of inliers with the contamination parameter (%) to know how many anomalies have to be added to the dataset
4. Draw that many molecules from the drugbank that do not contain any of the selected substructures, put them into the dataset and label them as anomalies

In order to describe those datasets, the nomenclature in Figure 4.2 is used. For each combination

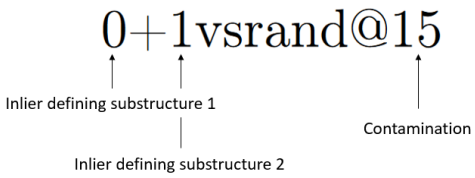

Figure 4.2: Nomenclature for the datasets

of two inlier defining substructures multiple synthetic datasets are created with varying contamination. Table 4.1 shows information about the datasets with a contamination of 15%. Notably, the average number of nodes for inliers is larger for every dataset. This is due to the necessary existence of a certain substructure.

Table 4.1: Datasets with a contamination of 15%

Name	# molecules	average # nodes inliers	average # nodes anomalies
0+1vsrand@15	1565	29.45	24.92
0+2vsrand@15	1362	30.41	22.65
0+3vsrand@15	2006	30.99	24.48
0+4vsrand@15	2481	29.65	25.87
0+5vsrand@15	1525	29.17	26.14
0+6vsrand@15	2447	32.34	25.05
1+2vsrand@15	1207	28.96	27.89
1+3vsrand@15	1390	30.43	25.37
1+4vsrand@15	1905	28.95	28.17
1+5vsrand@15	920	27.33	24.83
1+6vsrand@15	1920	32.79	24.05
2+3vsrand@15	1630	30.70	27.10
2+4vsrand@15	2140	29.28	27.35
2+5vsrand@15	1181	28.86	23.31
2+6vsrand@15	2152	32.62	24.02
3+4vsrand@15	2265	30.12	24.62
3+5vsrand@15	1330	30.31	23.95
3+6vsrand@15	1884	32.06	24.51
4+5vsrand@15	1868	28.84	26.99
4+6vsrand@15	2849	31.84	22.93
5+6vsrand@15	1773	33.26	26.88

4.1.4 Encoding

Whenever molecule fingerprints are used for the following experiments, the length of the fingerprints is set to 2048 and the radius is set to 2 because these are standard values and changing these parameters did not have a significant effect on the outcome of the experiments.

When encoding the synthetic datasets as graphs the node embedding included the information about the atomic number, the formal charge, the hybridization, whether the atom is part of an aromatic bond and whether the atom is part of a ring. Because the atomic number decides the element and that feature is categorical in nature, a one hot encoding of the atomic number is also added to the node embedding. The edge labels are ignored for the encoding because the hybridization and the aromaticity already contain information about the edge types and the edges would also have to be one hot encoded which would have required a very sophisticated GNN architecture.

4.2 Experimental setup

Three different types of experiments are performed in order to explore chemical anomaly detection. Firstly, classification methods are used to check the difficulty of synthetic datasets. This is done by using a Random Forest in combination with molecular fingerprints at different contamination levels and a binary GNN classifier with DiffPool layers.

Secondly, anomaly detection experiments are performed using classical methods of dimensionality reduction followed by standard anomaly detectors in combination with molecular fingerprints, an autoencoder whose loss is interpreted as an anomaly score with molecular fingerprints and the approach outlined in subsection 2.2.1 with max pooling DiffPool as aggregation module. These three approaches are chosen because each of them can be interpreted as following a different

conceptual approach outlined in section 3.2.

Finally, an experiment is constructed that includes knowledge about $\mathcal{P}$, namely that subgraphs are the determining factor for being an inlier and about the approximate size of the subgraphs of interest. This experiment performs gSpan on the dataset and then creates bit vectors for each molecule where each bit represents the presence or absence of a mined frequent subgraph. The number of frequent subgraphs present in each molecule is interpreted as an inlier score. This approach could be considered to belong to the conceptual approach of learning for feature extraction because each graph is reduced to a vector and with this information the anomaly score is calculated separately.

4.2.1 Classification

A random forest classifier and a GNN classifier are used for the classification experiments.

Random Forest Classifier

For the random forest classifier, the implementation of skLearn [39] is used together with the molecular fingerprint implementation of rdKit [40]. One hundred trees are used. 20% of the molecules are used as test data for every run and the Matthew's correlation coefficient (MCC) is used to evaluate the performance. Because these experiments are performed to evaluate the difficulty of the synthetic datasets, both inliers and anomalies are used in the training set.

GNN Classifier

For the experiment with GNN classifier, the following setup is used: 2 blocks each consisting of 2 graph convolution layers with a depth of 64 and a DiffPool layer followed by a graph convolution layer are used as an encoder and for the classification two regular dense layers are used with 64 nodes each. The sizes of n_i are chosen to accommodate the largest molecules and then quartering the graph size every aggregation step. Hereafter, a binary classifier with two fully connected layers with 64 nodes and binary cross entropy as classification loss term is used. The GNN classifier is implemented with PyTorch Geometric. Again, a mixture of anomalies and inliers is used for training and the MCC is used in order evaluate the performance. The decision boundary is determined by using the highest probability.

4.2.2 Anomaly detection

Classical methods, an autoencoder, and two GNN based, where one tries to improve upon the other by the use of DiffPool, approaches are used as anomaly detectors.

Dimensionality reduction and classical anomaly detector

In order to find the most suitable dimensionality reduction technique and classical anomaly detection algorithm for the task of chemical anomaly detection several preliminary experiments are carried out. The tested dimensionality reduction methods are PCA, MDS [39], and UMAP [41] and the tested anomaly detectors are local outlier factor (LoF), DBScan, OneClassSVM, and isolation forest [39]. Toy datasets with limited size are created by using a substructure to define inliers and a different substructure to define outliers. As described in chapter 1 this is more of a classification experiment but as classification is intuitively an easier problem than anomaly detection (learning decision boundary vs. learning underlying distribution), it should be still a valid approach for finding out which method is most suitable for chemical data.

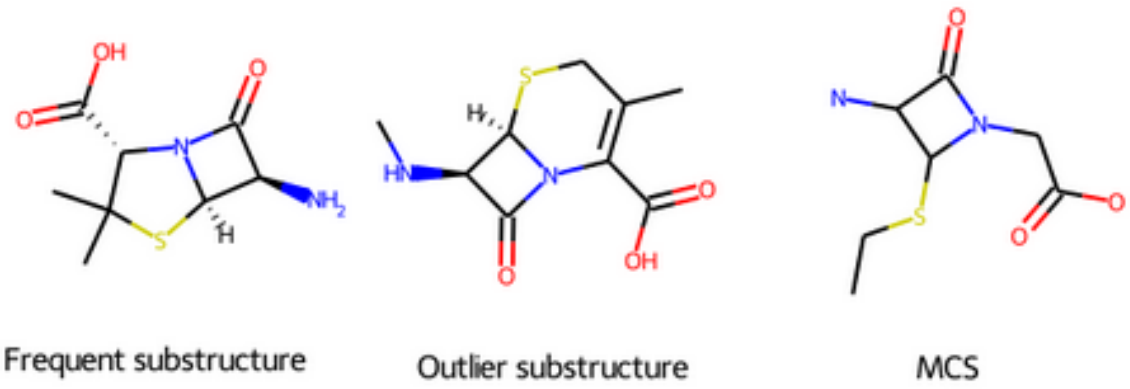

Figure 4.3: Left: Substructure defining inliers, middle: substructure defining outliers, right: maximal common substructure

In Figure 4.4 one can see the results of one of the preliminary experiments. In Figure 4.3 one can see the substructures chosen for defining the anomalies and the inliers. In total there are 40 inlier molecules and 3 outliers. For that specific experiment the standard parameters of the SKLEARN implementation of PCA and LoF are used, and the input is projected to 2 dimensions with PCA, and 5 neighbors are observed with LoF. At first glance these results might look promising

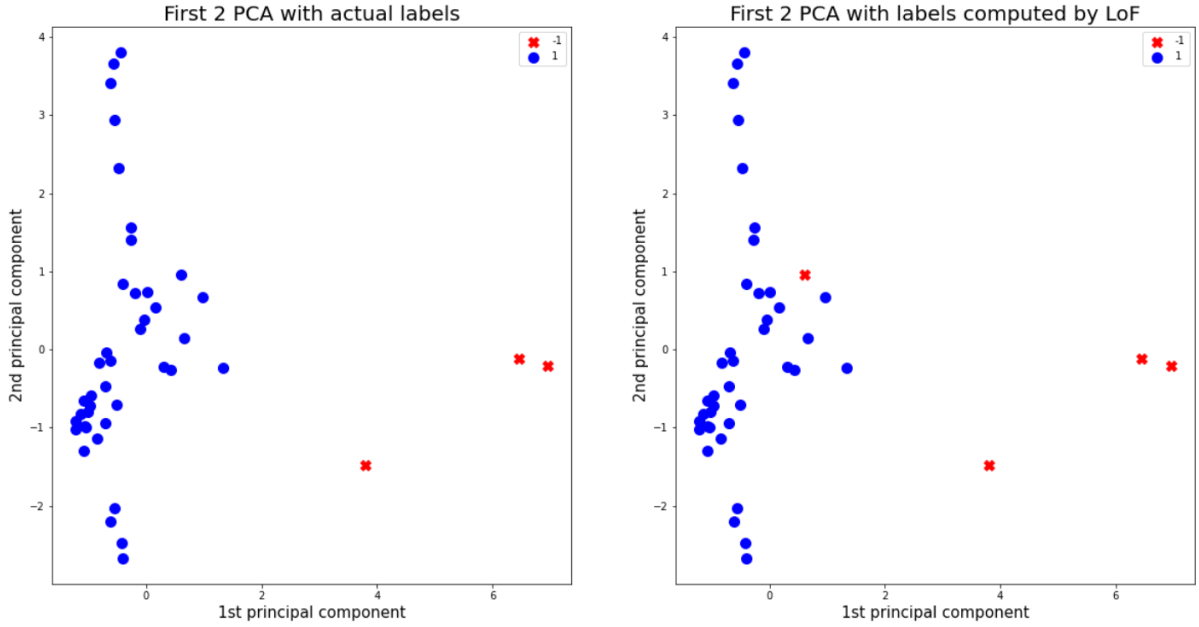

Figure 4.4: Example result from PCA with very few data points. On the left one can see the actual labels and, on the right, the calculated labels. -1 represents outliers and 1 represents inliers.

because the anomalies are detected and there is only false positive but when performing the same experiments on bigger datasets, the result looks totally different. In Figure 4.5 one can see what happens when the size of the data set is increased by using benzene as defining substructure for inliers and cyclohexane as defining substructure for anomalies such that the number of molecules in question is more like what one would expect in a real data set. In this case there are 6838 inliers and 3 anomalies. Only one of the 3 inliers is identified correctly and there are nearly 300 false positives.

As all combinations of anomaly detector and dimensionality reduction technique showed similar results, only the combination of PCA with LoF is investigated more in depth by applying the

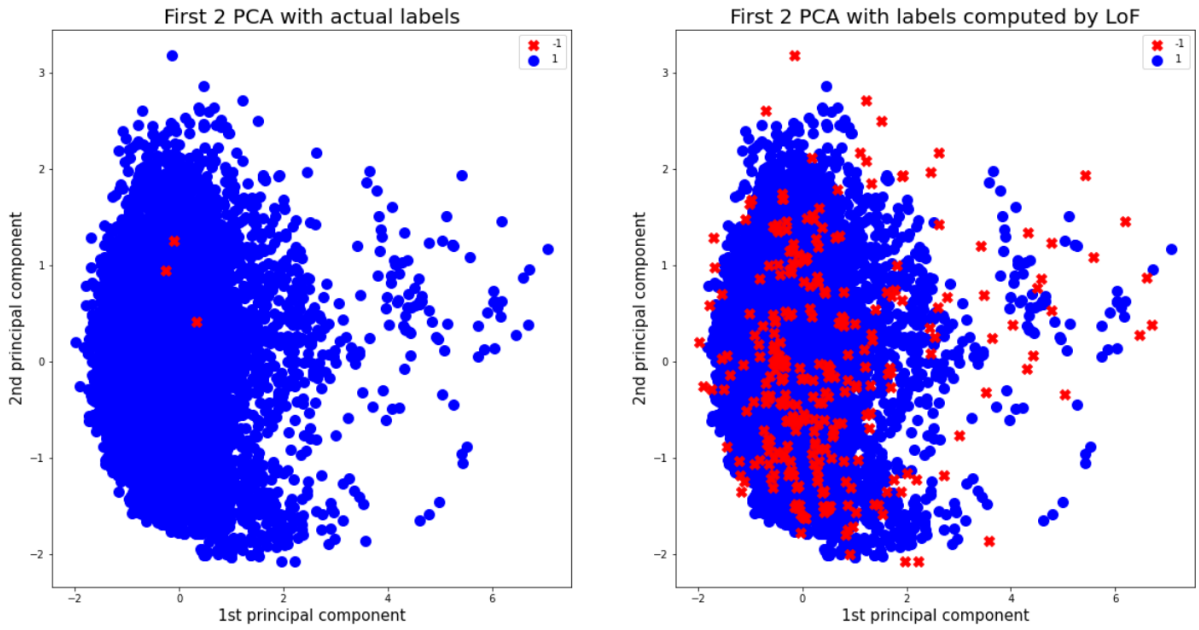

Figure 4.5: Example result from PCA with large amount of data. On the left one can see the actual labels and on the right the calculated labels. -1 represents outliers and 1 represents inliers.

pipeline (PCA as dimensionality reduction to two dimensions, followed by LoF with 5 observed neighbors) to the synthetic datasets with a contamination of 15%. The principle component analysis is trained on a subset of inlier and the test data is subsequently transformed and evaluated with the Local Outlier Factor.

Autoencoder with anomaly score from loss

An undercomplete autoencoder is implemented using the library PyTorch. The encoder has three layers decreasing the dimensionality from 2048 to 128 by first halving the size of the layer twice and then quartering it. The decoder starts with a layer with a size of 128 which is increased to 512, then 1024 and finally 2048. Other dimensionalities are also tested but the result remains the same. The layers are fully connected as convolutional operations do not make sense in combination with molecular fingerprints because neighboring bits do not form a more semantic meaning like it would be the case in images. For testing various loss function a set containing 3000 molecules with a benzene ring are used as inliers and 100 randomly selected molecules not containing this functional group are used as anomalies. The ADAM optimizer is used together with a learning rate of 0.001, 30 epochs and a batch size of 32.

Five loss function are tested for the autoencoder: the l_2-loss, the l_1-loss, the Tanimoto similarity, the cosine loss, and a custom loss, the weighted Tanimoto similarity. The weighted Tanimoto similarity is the Tanimoto similarity weighted with the inverse of the frequency of certain bits. The intention behind this idea is to make the less frequent bits, hopefully the ones activated by the anomalies, more important. When using the l_2 the sparsity of the input vectors is learned. This means that practically all values in the output vector are close to zero. The Tanimoto similarity seems to be a natural choice for a loss function because it is often used to compare molecules. For example, Casalegno M. et al. [2] use it to construct a similarity matrix. Whilst there is a trend for anomalies to have a higher reconstruction error/anomaly score, it is by no means significant. Additionally, the learnt representations are incorrect (manual checking if active bits remain active in output on a sample basis and comparing sums of input and output vector). When weighting the Tanimoto similarity the trend disappears, and the learnt representations are still incorrect. The l_1 loss exhibits an interesting behavior: All bits that are activated in more than half of the

compounds are constructed while the rest is not, and all other bits are close to zero. However, the reconstruction is still poor. The best reconstructions are achieved with the cosine loss. While the inactive bits are very close to zero, many active bits are definitely non-zero, but most are still closer to zero than to one.

Finally, the cosine loss is selected for further experimentation because its reconstruction is best compared with the other loss functions and in order to evaluate the performance of the autoencoder as an anomaly detector, it is trained with inlier molecules only and afterwards, the loss of each molecule in the test set is directly used as an anomaly score and the best decision boundary is selected from the AUC and the MCC is calculated to evaluate the performance.

GLOCAL anomaly detection with DIFFPOOL aggregation

A random target and a predictor network are implemented using PYTORCH GEOMETRIC. Both networks consist of two blocks of two graph convolution layers followed by a DIFFPOOL layer. The approach of by Ma R. et al. [28] is followed by only training the prediction network on only the inliers by comparing its output against the output of the random target network which distills information about the input graph via its structure which is encoded because the propagation through that network is dependent on the graph connectivity. Figure 3.1 depicts the architecture of the network. The aggregation module in this case is the DIFFPOOL layers. The local graph information is taken before the first DIFFPOOL module and the global graph information is taken in the end of the network. The depth of the convolutional layers is set to 64 and the number of nodes for the different pooling layers is equal to the number of atoms in the largest molecule initially and is then decreased by a fourth and finally by a fourth again. The learning rate is 0.0001 the batch size 32 and 30 epochs are run with the ADAM optimizer because it performs well in many scenarios. This setup is used on the AIDS dataset and the synthetic datasets with a contamination of 15%. Because of the different magnitudes of the entropy part of the DIFFPOOL loss, the link prediction loss term of DIFFPOOL and the loss coming from GLOCAL anomaly detection the losses are scaled such that they are similar in magnitude by multiplying the GLOCAL anomaly detection the loss by 10 and dividing the link prediction loss by 10^4. Only the entropy and link prediction loss of the predictor network are used for calculating the loss as the same terms from the random target network only change depending on the composition of the mini batch and not the current training progress.

GLOCAL anomaly detection

A random target and a predictor network are implemented using PYTORCH GEOMETRIC. Both networks consist of four layers of graph convolution layers with an initial depth of 512 and only in the last layer a depth of 256. These values are chosen to be the same as in the paper from Ma R. et al. [28]. In order to test the correctness of the approach, the AIDS dataset is used. All experiments are performed with a batch size of 32, a learning rate of 0.0001 with the Adam optimizer over 30 epochs. As pooling operation max neighbor pooling is chosen.

4.2.3 Anomaly detection with prior knowledge

If there is some prior knowledge about the type of property $\mathcal{P}$ that defines the inliers, then the algorithm $\mathcal{F}$ that tries to approximate $\mathcal{P}$ can be tailored to the problem. For example, the way that the synthetic datasets are created leads naturally to frequent subgraph mining because it is to be expected that the subgraphs that define the inliers are among the biggest frequent subgraphs. Therefore, performing FSM on the training set with the requirement of the frequent subgraphs having a certain size, followed by counting how many of the frequent subgraphs can be found in every molecule leads to a convenient way of finding anomalies because anomalies should in general

contain few of the frequent subgraphs. The aforementioned procedure is followed with gSpan as an FSM algorithm to find the frequent subgraphs in the synthetic datasets with a contamination of 15 % given a certain support and minimal size for frequent subgraphs. Afterwards, the frequent subgraphs are counted for each molecule and the count is interpreted as an inlier score (high is good). The AUC and subsequently the MCC is calculated. The minimal support is decided to be 400 and the size of the interesting subgraphs is decided to be between 6 to 10 in order to test if performance is better for datasets which are determined by subgraphs in that range.

5 | Results

The aim of this chapter is to show that the synthetic datasets can be classified with there are sufficient anomalies for the training set, to show that the selected anomaly detection methods that are described in subsection 4.2.2 fail to properly predict anomalies, and that there is an anomaly detection algorithm is able to solve the synthetic datasets, which is described in subsection 4.2.3

5.1 Classification

In order to ensure that algorithms can be trained that are able to distinguish between inliers and anomalies, two classifiers are trained. The first classifier is a random forest classifier that uses fingerprints as input, and it is chosen to show that fingerprints contain enough information for deciding whether a test molecule is an inlier or an anomaly. A GNN classifier is trained to show that the graph encoding of molecules is sufficient for classification for the datasets. For both experiments the contamination is varied in order to show that a significant portion of the known molecules have to be anomalies in order to draw a good decision boundary.

5.1.1 Random Forest Classifier

A Random Forest classifier is trained on the synthetic data sets while varying the contamination. The evaluation is measured with the Matthews correlation coefficient. Figure 5.1 shows the results of the experiments with the random forest classifier. Increasing the contaminations leads to MCC being close to 0.8 for all datasets. The sporadic behavior at low contamination for many datasets is notable. This can be explained by the fact that even guessing few anomalous compounds correctly, leads to a huge improvement in the MCC as it is designed to evaluate the performance of classification algorithms of unbalanced datasets. Further, the difference between the experiments with the same inlier defining substructures with varying contaminations is very large. This can be explained by the fact the anomalous compounds are chosen anew for every dataset. This means that it is possible that the compounds with a contamination of 4% might be much easier to identify than the compounds with a contamination of 10%. Notably, all of the synthetic datasets are solvable with a sufficient contamination with a random forest classifier as indicated by the high MCC. This is likely due to the fact that there is some bit in the molecular fingerprints that is always turned on for inliers and always turned off for anomalies but the algorithm is unable to find that bit with lower contaminations because there are other bits by chance that explain the difference between inlier and anomalies in the training set better than the actual inlier determining bit.

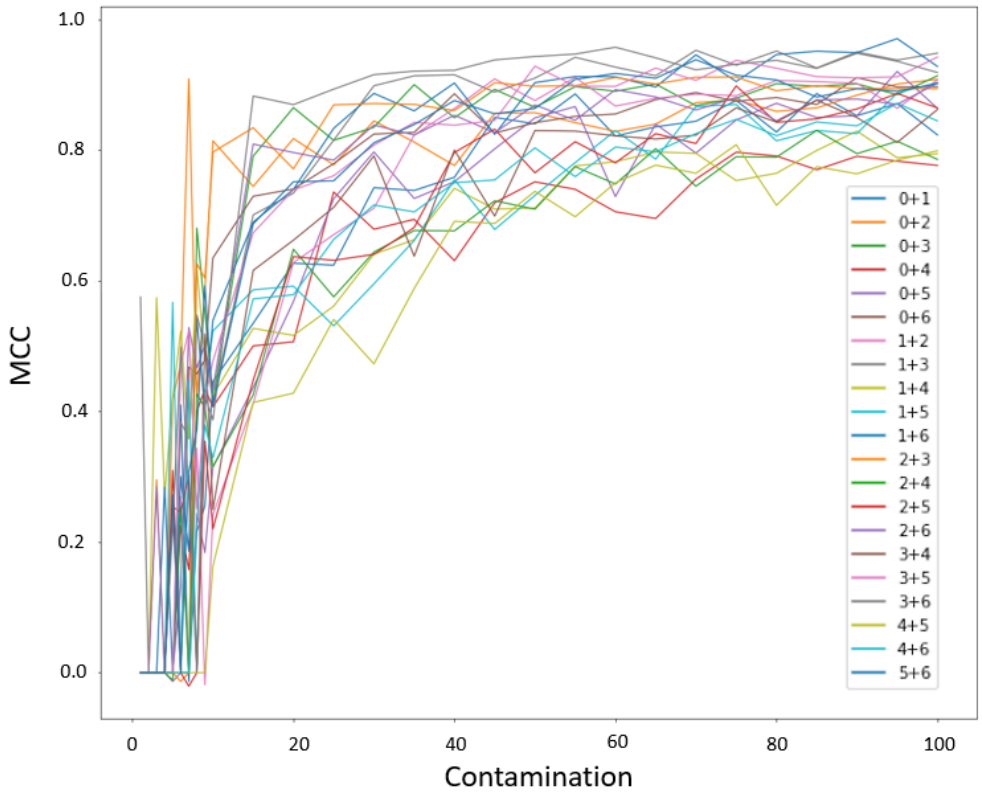

Figure 5.1: Results of the random forest classifier

5.1.2 GNN classifier

A GNN classifier is trained on the synthetic data sets while varying the contamination. The evaluation is measured with the Matthews correlation coefficient. Figure 5.2 shows the results of the classification experiments with the graph neural network. The classification fails below 20% anomaly contamination. Just like in the experiments with the random forest classifier the difference between consecutive experiments is large because the anomalies are picked randomly again when increasing the contamination. At 100 % contamination the MCC averages 0.47 but the variance is huge with 0.17. There is even the dataset "1+5vsrand@100" which scores a MCC of 0.00 meaning that the classification does not work at all. The poor performance can be explained by the fact that both inlier defining substructures contain a carboxylic acid group. This moiety is very common and therefore there are also a lot of anomalies that contain this group. Nonetheless, the highest MCC for every combination of inlier defining substructures is achieved at contaminations greater or equal to 80 except in the combination of substructure 2 and substructure 6 which has the highest MCC (0.82) at 65% contamination. The inability to classify at low contaminations can be attributed to the GNN learning that it is optimal to always guess that the input is an inlier because there are so many inliers in the training set. This leads to a local minimum for always returning the inlier label, which cannot be escaped.

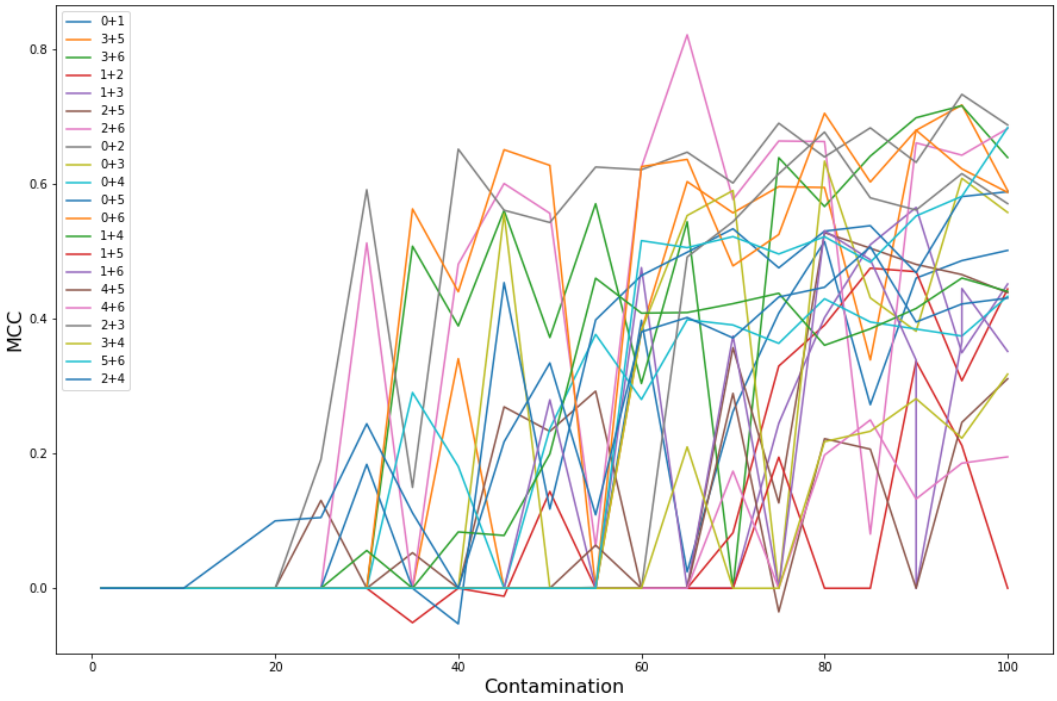

Figure 5.2: Results of the GNN classifier

5.2 Anomaly detection

Table 5.1 shows the resulting MCCs for all anomaly detection experiments. MCC_{class} are the experiments for which PCA followed by Local Outlier factor is used for anomaly detection, MCC_{AE} are the experiments with the autoencoder, MCC_{GCN} are the experiments based on the GLOCAL anomaly detection with max pooling, and $MCC_{DiffPool}$ shows with results of the same experiments but with DIFFPOOL as pooling modules. Finally, MCC_{gSpan} is the anomaly detection algorithm that is described in subsection 4.2.3. Clearly, all anomaly detection techniques fail on the synthetic data sets except the FSM-based one.

5.2.1 Dimensionality reduction and classical anomaly detector

Table 5.1 shows the results of the experiments with principal component analysis and local outlier factor. It is evident that this method is unable to perform anomaly detection properly because the MCCs are close to zero. When looking at PCA in detail it becomes immediately apparent why these dimensionality reduction techniques failed to perform anomaly detection on bigger datasets: In order to preserve 90% of the information, one would have to use around 98% of dimensions. Therefore, PCA is not a suitable technique for reducing the dimensionality in this context.

5.2.2 Autoencoder with anomaly score from loss

Table 5.1 shows the results of the long experiment with the autoencoder with the cosine loss. It is evident that this method is unable to perform anomaly detection properly because the MCCs are close to zero. These results are not surprising as the input is hardly reconstructed with this approach. Figure 5.3 shows a typical histogram for the losses of inliers (blue) and the anomalies (orange). This also explains why the MCCs are negative: the anomalies have a lower anomaly score (loss) on average than the inliers. Therefore, there is no decision boundary that is able to find the anomalies correctly. Maybe, this is caused by the inliers having a substructure that

Table 5.1: Anomaly detection results for the datasets with a contamination of 15% color coded by success

Name	MCC_{class}	MCC_{AE}	MCC_{GCN}	$MCC_{DiffPool}$	MCC_{gSpan}
0+1vsrand@15	-0.02	-0.06	0.22	-0.16	0.49
0+2vsrand@15	-0.16	-0.05	0.14	-0.20	0.78
0+3vsrand@15	0.03	-0.12	-0.25	-0.14	0.49
0+4vsrand@15	-0.13	-0.01	0.23	-0.17	0.48
0+5vsrand@15	-0.09	-0.14	0.18	-0.14	0.52
0+6vsrand@15	-0.11	-0.13	-0.03	-0.07	0.63
1+2vsrand@15	-0.04	-0.14	0.12	0.04	0.50
1+3vsrand@15	-0.01	-0.16	-0.35	-0.05	0.09
1+4vsrand@15	-0.17	-0.08	-0.23	-0.12	0.52
1+5vsrand@15	-0.12	-0.11	0.09	0.22	0.19
1+6vsrand@15	-0.14	-0.12	-0.16	-0.09	0.71
2+3vsrand@15	-0.01	-0.16	-0.28	-0.12	0.61
2+4vsrand@15	-0.01	-0.00	0.16	-0.08	0.36
2+5vsrand@15	-0.07	-0.13	0.15	-0.20	0.42
2+6vsrand@15	-0.14	-0.14	-0.04	-0.16	0.13
3+4vsrand@15	-0.06	-0.03	-0.19	-0.14	0.34
3+5vsrand@15	-0.10	-0.18	-0.38	-0.10	0.54
3+6vsrand@15	-0.05	-0.18	-0.40	-0.10	0.66
4+5vsrand@15	-0.01	-0.04	0.10	-0.11	0.37
4+6vsrand@15	-0.02	-0.00	-0.09	-0.19	0.52
5+6vsrand@15	-0.11	-0.15	-0.16	-0.08	0.75

defines them leading to larger molecules on average (Table 4.1), leading to more activated bits in the fingerprints and finally a higher loss because the main feature of fingerprints that is learned is their sparsity. Maybe, it is possible to improve the reconstruction by adding additional terms like entropy term, similar to what is used in DiffPool, to the loss function or training with some kind of entropy-based loss.

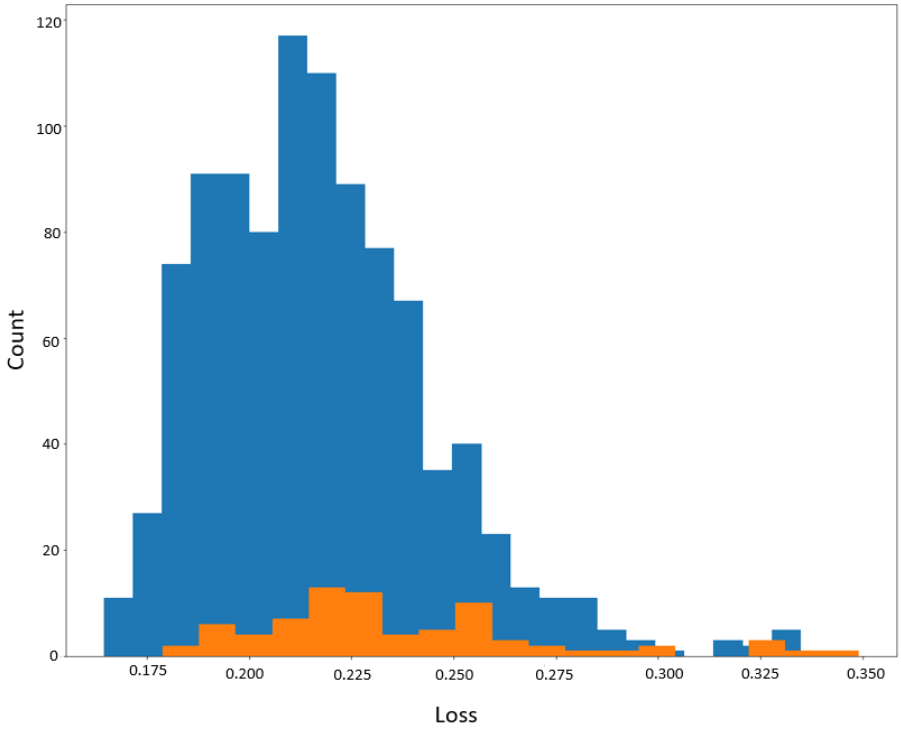

Figure 5.3: Typical histogram for the loss of the anomalies and the inliers

5.2.3 GLOCAL anomaly detection

Figure 5.4 shows a typical learning curve for an experiment with the anomaly detection technique outlined in subsection 2.2.1 with max pooling as aggregation module. The kink that breaks the plateau at around mini batch 500 indicates that some important characteristic is learned. This indicates that there might be room for improvement with more careful learning rate tuning and possibly a learning rate scheduler and longer training but this behavior does not occur in every loss curve. Table 5.1 shows the results with the synthetic datasets. When performing the experiment with the AIDS dataset the MCC is 0.98 and hence anomaly detection works extremely well as outlined in the paper, but when taking a closer look at the number of nodes in the anomalies and the number of nodes in the inliers it becomes apparent why. Figure 5.5 shows a histogram of how many nodes the inliers and the anomalies have of the AIDS dataset have. If a molecule has more than 13 nodes, it is an anomaly and else it is an inlier. As can be seen from the table, anomaly detection on the synthetic dataset is not very successful. In many cases the anomalies even have a lower anomaly score than the inliers as indicated by the negative values. Because of the number of nodes dependent results of the AIDS dataset, the correlation between the difference between the average number of nodes of inliers and anomalies and the MCC is calculated. This relation is visualized in Figure 5.6. Interestingly, there is a negative correlation ($r = -0.51$) which means that if the average number of nodes in the anomaly and the inlier test set is close then the MCC is high and if the difference is big then the MCC is small and more often than not negative, which means that the significantly larger molecules of the inlier datasets are labeled as anomalies. Therefore, it seems that if the number of atoms one set, anomalies or inliers, is significantly larger than in the other set, then the larger molecules are assigned significantly higher anomaly scores. If the sets have approximately the same average number of nodes, then the desired effect occurs that anomalies are not as well encoded by the predictor network with regard to the random target network as inliers.

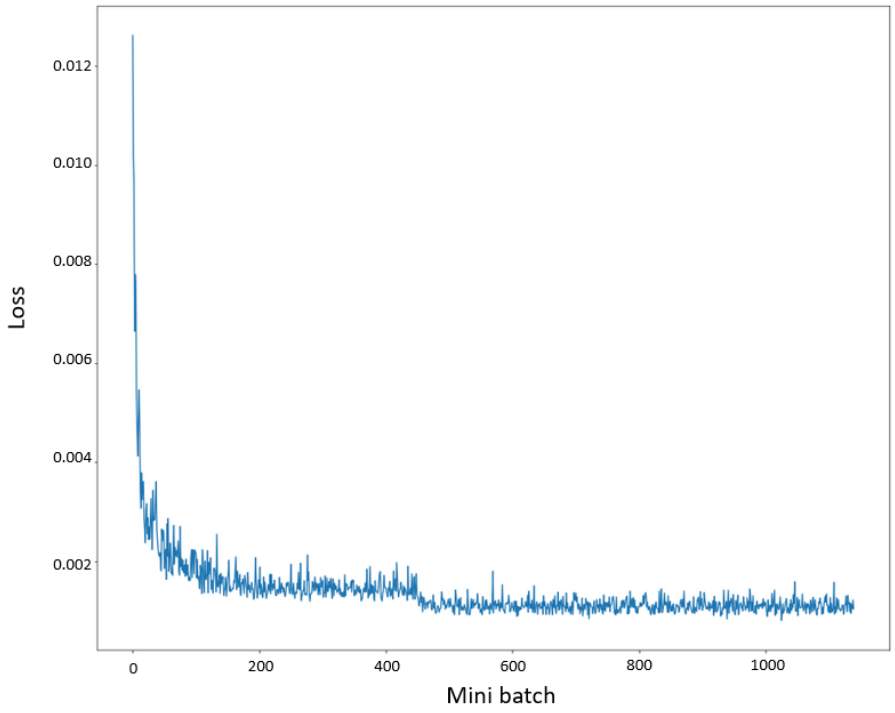

Figure 5.4: Selected loss curve for GLOCAL anomaly detection with max pooling

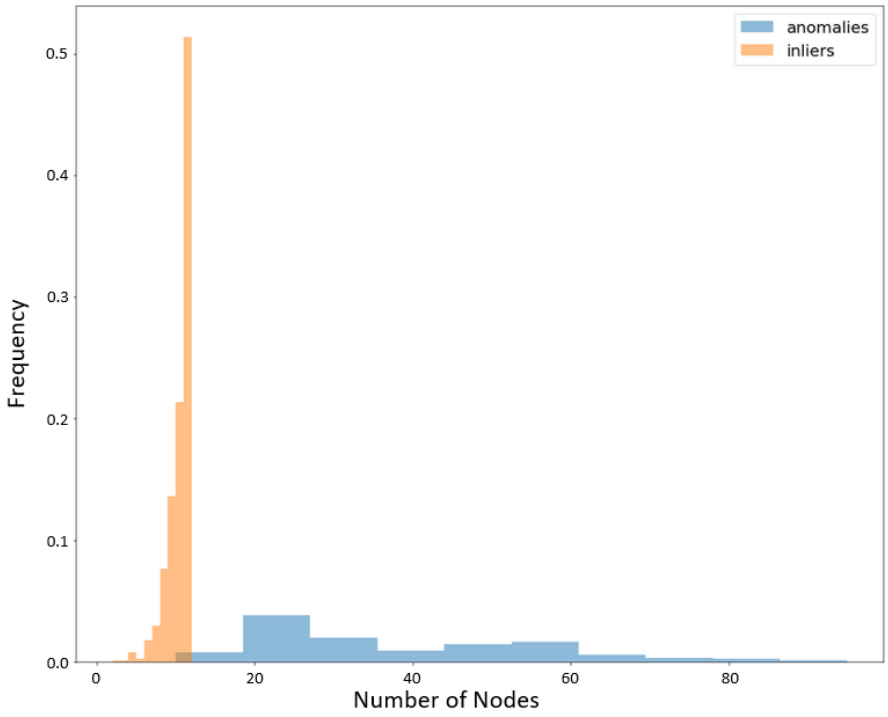

Figure 5.5: Histogram of the number of nodes of the AIDS dataset

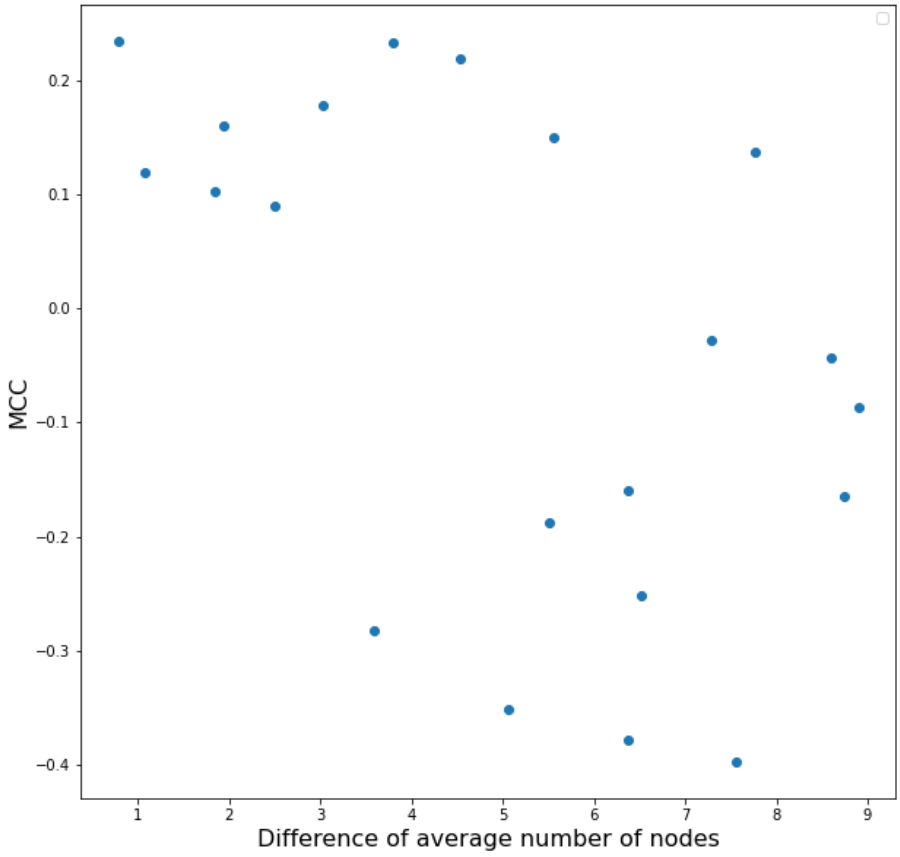

Figure 5.6: Relation between the difference of average number of nodes between anomalies and inlier and the MCC

5.2.4 GLOCAL anomaly detection with DiffPool aggregation

Figure 5.7 shows a typical learning curve for an experiment with the anomaly detection technique outlined in subsection 2.2.1 with DIFFPOOL as aggregation module. Notably, this curve is more spiky than the curve for the GLOCAL anomaly detection with max pooling and initially this was thought to be caused by above average number of nodes in a mini batch, but this is also observed in the AIDS dataset in which the average number of nodes stays relatively constant during training with around 11 nodes (Figure 5.5, inliers). For the AIDS dataset a MCC of around 0.91 is scored. Table 5.1 shows the results with the synthetic datasets. This experiment seems to have a similar problem to what is observed in GLOCAL anomaly detection with max pooling but the effect seems less pronounced because there is an experiment which resulted in a MCC with an absolute value of more than 0.22. There are only two entries that have a positive MCC. In both cases the initial graph size is comparatively small with 158 and 163 nodes compared to an average of 425 nodes. The graph size is also small for "2+5vsrand@15" but in the case of this dataset the difference between average number of atoms in inlier and anomaly set is relatively big with more than 5. With the notion of small differences between the average number of atoms and small maximal molecule sizes (which are equivalent to the graph sizes in the GNN), the dataset "1+2vsrand@15" should perform relatively well, but with a MCC of 0.04 this dataset does not indicate that this technique is suited well to be an anomaly detector as described in this context.

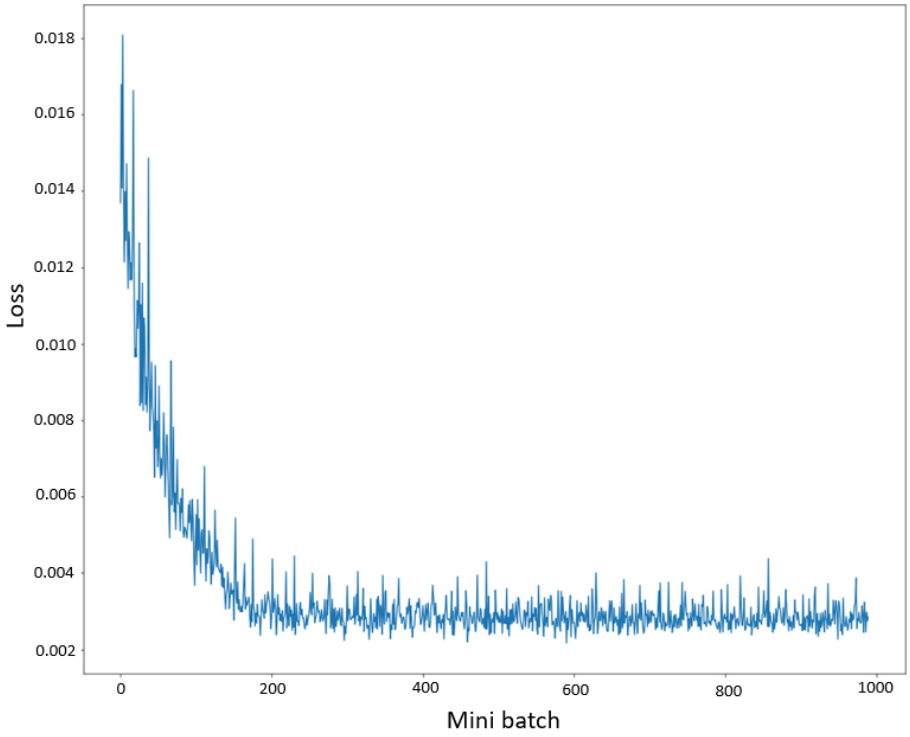

Figure 5.7: Typical loss curve for GLOCAL anomaly detection with DIFFPOOL

5.3 Anomaly detection with prior knowledge

An anomaly detector is trained based on the FSM based approach outlined in subsection 4.2.3. Table 5.1 shows the results with the synthetic datasets. For almost all of the datasets the anomaly detection worked well the exceptions being "1+3vsrand@15" with a MCC of 0.09, "1+5vsrand@15" with a MCC of 0.19, and "2+6vsrand@15" with a MCC of 0.13. Notably, all of these datasets have an inlier defining substructure that does not correspond to the criteria of having between 6 and

10 nodes for being counted as a frequent subgraph. Furthermore, it is noteworthy that the largest substructure, number 1, is contained in 2 of the datasets that perform worst. It is a possibility that this occurs due to the anomaly score assignment based on the number frequent substructures in each molecule, and due to the range of subgraphs that are accepted as frequent. For example, in a dataset with benzene as an inlier defining substructure it is highly likely that most of the inliers contain toluene because in many cases the benzene ring is connected to a carbon. This is also true for other substructures that contain benzene. All identified subgraphs that contain the inlier defining substructure contribute to the prediction power of the algorithm and because there is an upper limit for the size of frequent subgraphs, sets with large inlier defining substructures have likely fewer of these prediction power increasing subgraphs.

It is also possible that only part of the defining substructure occurs in the set of frequent subgraphs and is therefore used for detecting anomalies. These subgraphs can even be chemically invalid, for example, for benzene "Cccccc" is a subgraph that is likely frequent and chemically nonsensical. Therefore, an inlier is not only identified by its actual defining property but also by a plethora of substructures that are similar to the defining property.

But all frequent subgraphs that do not contain the entire inlier defining substructure can lead to false positives because their existence does not guarantee being an inlier, but they are still mined. Furthermore, larger inlier defining substructures lead to more frequent subgraphs that do not contain the inlier defining substructure because of the downward closure property.

5.4 Discussion of results

Twenty-one datasets which can be used to test anomaly detection algorithms are compiled. For high contaminations, state-of-the art classification techniques are able to differentiate between normal and abnormal instances very well. On the other hand, for low contaminations they still struggle. Furthermore, three representative anomaly detection techniques are unable to solve the anomaly detection problem, where only inliers are available for training. These results seem to indicate that a substantial number of anomalies is required for training an algorithm that is able to solve the anomaly detection problem with the compiled datasets. At the same time the notion of knowing anomalies during training goes against the problem of unknowness in anomaly detection. However, if prior knowledge is used to construct an anomaly detector, the datasets are solved with ease. This indicates that the core problem is not information provided to the algorithm but rather the feature extraction of the algorithm. This also explains the behavior of the random forest sometimes performing very well on low contaminations: if the features representing the inlier defining substructures are selected because they contain the highest degree of information for the majority of decision trees, then the MCC if very high. However, if there are features that have a higher information gain in the training set, then it is very likely that the classification is incorrect for the test data.

6 | Conclusion

In conclusion, twenty-one datasets which can be used to test anomaly detection algorithms are compiled based on an inlier defining property. With these datasets, three different types of experiments are performed. Firstly, a random forest classifier and a GNN classifier are used to show that classification is feasible with the datasets and that drawing a decision boundary between inliers and anomalies is possible if sufficient information is available about anomalies. Secondly, a classical, a deep, and a graph-level anomaly detection approach are selected, and experiments are performed that show that these techniques which follow different anomaly detection paradigms, fail to detect anomalies. Furthermore, the GNN-based approach is also tested with a more powerful pooling operation. Finally, an anomaly detection technique is constructed that is able to solve the dataset in order to show that anomaly detection is feasible with these datasets and that feature extraction might be the problem.

Based on this result, a more meaningful feature extraction method is required. Maybe, it is possible to use a pretrained network of some kind to ensure that the anomaly detection algorithm only takes important features into consideration. For example, instead of feeding a molecular graph directly into the random target network and predictor network, a pretrained encoder could be used create a graph representation of each molecule, whose features are more semantic in meaning. This network architecture could be trained end-to-end or the pretrained encoder could just be used for embedding the graphs.

Bibliography

[1] Guansong Pang, Chunhua Shen, Longbing Cao, and Anton Van Den Hengel. Deep learning for anomaly detection: A review. *ACM Comput. Surv.*, 54(2):1–38, 2021.

[2] Mosè Casalegno, Guido Sello, and Emilio Benfenati. Definition and detection of outliers in chemical space. *Journal of Chemical Information and Modeling*, 48(8):1592–1601, 2008.

[3] Tatiana I. Netzeva, Andrew Worth, Tom Aldenberg, Romualdo Benigni, Mark T. D. Cronin, Paolo Gramatica, Joanna S. Jaworska, Scott Kahn, Gilles Klopman, Carol A. Marchant, Glenn Myatt, Nina Nikolova-Jeliazkova, Grace Y. Patlewicz, Roger Perkins, David Roberts, Terry Schultz, David W. Stanton, Johannes J. M. van de Sandt, Weida Tong, Gilman Veith, and Chihae Yang. Current status of methods for defining the applicability domain of (quantitative) structure-activity relationships. the report and recommendations of ECVAM workshop 52. *Altern Lab Anim*, 33(2):155–173, 2005.

[4] William L. Jorgensen. The many roles of computation in drug discovery. *Science*, 303(5665):1813–1818, 2004.

[5] Vijay K. Gombar and Kurt Enslein. Assessment of n-octanolwater partition coefficient: When is the assessment reliable? *Journal of Chemical Information and Computer Sciences*, 36(6):1127–1134, 1996.

[6] Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly detection: A survey. *ACM Comput. Surv.*, 41(3):1–58, 2009.

[7] Guilherme Campos, Arthur Zimek, Joerg Sander, Ricardo Campello, Barbora Micenková, Erich Schubert, Ira Assent, and Michael Houle. On the evaluation of unsupervised outlier detection: measures, datasets, and an empirical study. *Data Mining and Knowledge Discovery*, 30(4):891–927, 2016.

[8] Longbing Cao. Coupling learning of complex interactions. *Information Processing &amp Management*, 51(2):167–186, 2015.

[9] Leman Akoglu, Hanghang Tong, and Danai Koutra. Graph-based anomaly detection and description: A survey. *Data Mining and Knowledge Discovery*, 29(3):626–688, 2015.

[10] Chen Qiu, Marius Kloft, Stephan Mandt, and Maja Rudolph. Raising the bar in graph-level anomaly detection. In *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI-22*, pages 2196–2203. International Joint Conferences on Artificial Intelligence Organization, 2022.

[11] Gurkamal Deol. Predicting carcinogens with machine learning and molecular fingerprinting. https://miro.medium.com/max/788/1*GYPRBWpHg6o5WvAyleLQhw.png. Accessed: 2022-09-19.

[12] Jie Zhou, Ganqu Cui, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, and Maosong Sun. Graph neural networks: A review of methods and applications. *AI Open*, 1:57–81, 2020.

[13] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*, 2017.

[14] Rex Ying, Jiaxuan You, Christopher Morris, Xiang Ren, William L. Hamilton, and Jure Leskovec. Hierarchical graph representation learning with differentiable pooling. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, page 4805–4815, 2018.

[15] Weikuan Jia, Meili Sun, Jian Lian, and Sujuan Hou. Feature dimensionality reduction: a review. *Complex & Intelligent Systems*, 8(3):2663–2693, 2022.

[16] Ian T. Jolliffe and Jorge Cadima. Principal component analysis: a review and recent developments. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 374(2065), 2016.

[17] Dor Bank, Noam Koenigstein, and Raja Giryes. Autoencoders. *CoRR*, 2020.

[18] Charu C. Aggarwal. *Outlier analysis*. Springer, New York, 2013.

[19] Srikanth Thudumu, Philip Branch, Jiong Jin, and Jugdutt Singh. A comprehensive survey of anomaly detection techniques for high dimensional big data. *Journal of Big Data*, 7(1), 2020.

[20] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proc. of 2nd International Conference on Knowledge Discovery and Data Mining*, pages 226–231, 1996.

[21] Markus Breunig, Peer Kröger, Raymond Ng, and Joerg Sander. Lof: Identifying density-based local outliers. *ACM Sigmod Record*, 29(2):93–104, 2000.

[22] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. Isolation forest. In *2008 Eighth IEEE International Conference on Data Mining*, pages 413–422, 2008.

[23] David M. J. Tax and Robert P. W. Duin. Support vector data description. *Machine Learning*, 54(1):45–66, 2004.

[24] Chuntao Jiang, Frans Coenen, and Michele Zito. A survey of frequent subgraph mining algorithms. *The Knowledge Engineering Review*, 28(1):75–105, 2013.

[25] X. Yan and J. Han. gspan: Graph-based substructure pattern mining. Technical report, University of Illinois at Urbana-Champaign, Department of Computer Science., 2002.

[26] Michihiro Kuramochi and George Karypis. Frequent subgraph discovery. In *Proceedings 2001 IEEE International Conference on Data Mining*, pages 313–320, 2001.

[27] X. Yan and J. Han. gspan: Graph-based substructure pattern mining. In *Proc. 2002 of Int. Conf. on Data Mining*, pages 721–724, 2002.

[28] Rongrong Ma, Guansong Pang, Ling Chen, and Anton van den Hengel. Deep graph-level anomaly detection by glocal knowledge distillation. In *Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining*, page 704–714, 2021.

[29] Yuri Burda, Harrison Edwards, Amos J. Storkey, and Oleg Klimov. Exploration by random network distillation. *CoRR*, 2018.

[30] Lingxiao Zhao and Leman Akoglu. On using classification datasets to evaluate graph outlier detection: Peculiar observations and new insights. *Big Data*, 2021.

[31] Paul D. Dobson and Andrew J. Doig. Distinguishing enzyme structures from non-enzymes without alignments. *Journal of Molecular Biology*, 330(4):771–783, 2003.

[32] Karsten M. Borgwardt, Cheng Soon Ong, Stefan Schönauer, S. V. N. Vishwanathan, Alex J. Smola, and Hans-Peter Kriegel. Protein function prediction via graph kernels. *Bioinformatics*, 21(Suppl 1):i47–i56, 2005.

[33] Ida Schomburg, Antje Chang, Christian Ebeling, Marion Gremse, Christian Heldt, Gregor Huhn, and Dietmar Schomburg. Brenda, the enzyme database: updates and major new developments. *Nucleic Acids Res.*, 32(Database issue):D431–3, 2004.

[34] Kaspar Riesen and Horst Bunke. Iam graph database repository for graph based pattern recognition and machine learning. In *Structural, Syntactic, and Statistical Pattern Recognition*, pages 287–297, 2008.

[35] Nikil Wale, Ian A. Watson, and George Karypis. Comparison of descriptor spaces for chemical compound retrieval and classification. *Knowledge and Information Systems*, 14(3):347–375, 2008.

[36] Pinar Yanardag and S.V.N. Vishwanathan. Deep graph kernels. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, page 1365–1374, 2015.

[37] Christopher Morris, Nils M. Kriege, Franka Bause, Kristian Kersting, Petra Mutzel, and Marion Neumann. Tudataset: A collection of benchmark datasets for learning with graphs. In *ICML 2020 Workshop on Graph Representation Learning and Beyond (GRL+)*, 2020.

[38] David S. Wishart, Yannick D. Feunang, An C. Guo, Elvis J. Lo, Ana Marcu, Jason R. Grant, Tanvir Sajed, Daniel Johnson, Carin Li, Zinat Sayeeda, Nazanin Assempour, Ithayavani Iynkkaran, Yifeng Liu, Adam Maciejewski, Nicola Gale, Alex Wilson, Lucy Chin, Ryan Cummings, Diana Le, Allison Pon, Craig Knox, and Michael Wilson. Drugbank 5.0: a major update to the drugbank database for 2018. *Nucleic Acids Res*, 46(1):1074–1082, 2018.

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

[40] Greg Landrum. Rdkit: Open-source cheminformatics software. https://www.rdkit.org/.

[41] Mario Lovrić, Tomislav Đuričić, Han T. N. Tran, Hussain Hussain, Emanuel Lacić, Morten A. Rasmussen, and Roman Kern. Should we embed in chemistry? a comparison of unsupervised transfer learning with pca, umap, and vae on molecular fingerprints. *Pharmaceuticals*, 14(8), 2021.