Graph Representation Through Topological Descriptors

Bachelor Thesis

Faculty of Science, University of Bern

submitted by

Robin Gibson

from Zug, Switzerland

Supervision:

PD Dr. Kaspar Riesen

Mathias Fuchs

Institute of Computer Science (INF)

University of Bern, Switzerland

Abstract

Graphs offer the advantage of representing data with binary relations and without fixed sizes, such as molecules. The downside of using them is that they introduce a higher complexity which makes tasks like binary classification less efficient. Their counterpart in data representation, the feature vector, allows for faster computation time but lacks the two mentioned benefits. This thesis explores the topological descriptors embedding (TDE), a method that maps a graph to a feature vector while retaining as much of the structural properties of the graph as possible. This process enables the use of more efficient classification algorithms on graphs. In this thesis, we employ fourteen topological descriptors and three label descriptors to perform the TDE on eight different data sets containing chemical compounds. We use three different binary classifiers (k-NN, SVM, ANN) in combination with the TDE and compare the prediction accuracies to three respective reference systems. The k-NN and SVM use the graph edit distance, while the ANN is compared to a GCN. We achieve competitive or better accuracies when pairing the TDE with the SVM or k-NN. Conversely, it underperforms when combined with the ANN. Additionally, we track selected features when applying two feature selection methods to the TDE and find that certain features seem to be beneficial across all the data sets.

Acknowledgements

I am deeply grateful to PD Dr. Kaspar Riesen and Mathias Fuchs for not only supervising this thesis but also for providing the key idea that set this work in motion. I highly appreciate Dr. Riesen's extensive expertise and knowledge in the field, which have been invaluable to the shaping of this thesis. Special thanks go to Mathias Fuchs for his thorough guidance and thoughtful advice, as well as his consistent availability at every stage of this thesis.

Contents

Chapter 1

Introduction

This bachelor's thesis lies in the field of artificial intelligence (AI). One way of defining AI is: "Artificial intelligence (AI) is the field of research that strives to understand, design, and build cognitive systems. From computer programs that can beat top international grand masters at chess to robots that can help detect improvised explosive devices in war, AI has had many successes." [1] The first program described as an artificial intelligence program was called the Logic Theorist and was written in 1956 [2]. The program was able to prove mathematical theorems just like a mathematician could. Since then AI has made significant progress and for a more detailed overview, see the review by Brunette [3]. There are several uses for AI. For example, recommendation algorithms are used by Amazon or Netflix to provide personalized recommendations of items [4], and facial expression recognition algorithms can be used for surveillance or driver safety [5].

One of many sub-fields embedded in AI is machine learning (ML). Machine learning uses data, features, and different models to perform different tasks. In essence, the goal of the ML approach is to map outputs as correctly as possible to data points (i.e. perform predictions). Classification and regression are two examples among numerous ML tasks designed to accomplish this goal. A common task in machine learning is binary classification, where one of two possible classes is mapped to each data point. A practical example could be the classification of whether an email is spam or not. Various models that perform this type of classification exist, such as the k-nearest neighbors model (k-NN), the support vector machine (SVM), and the artificial neural network (ANN), just to name a few. The classification models are trained on training data and can then be used to perform predictions on new data [6].

This ML approach can also be adopted for pattern recognition (PR), a subfield of ML. Pattern recognition aims to automatically discover regularities in data. It uses computer algorithms to find them and then performs tasks (e.g. classification), based on the identified patterns [7]. The two main approaches to solving the classification task in PR are the *statistical* and the *structural* approaches [8].

The following paragraph summarizes the different approaches and is based on the paper by Bunke et al. [8]. In the statistical approach, the patterns are represented as feature vectors which are formally represented as n-dimensional vectors in the feature space $\mathbb{R}^n$. This approach offers the advantage of the availability of many mathematical operations which can be applied in the real coordinate space. Due to this, numerous classification algorithms with low computational complexity are available. The two main disadvantages of the statistical approach are that feature vectors do not possess the ability to describe binary relations and that they are constrained to a fixed size. These disadvantages do not apply to the structural approach. In the structural approach, the data is represented as graphs. Graphs are composed of nodes and edges (which form a binary connection between a pair of nodes) and can vary in size and simplicity. Consequently, the two previously mentioned limitations do not exist. However, along with the representational power gained from using graphs also comes a higher computational complexity. This is because some basic mathematical operations like summation or multiplication do not exist between graphs. Even checking for equality between two graphs is computationally a lot more expensive than checking whether two vectors are equal. Two common approaches that mitigate the increased computational complexity when using graphs are the application of *graph kernels* or *graph embeddings*.

The method of using a *topological descriptors embedding* (TDE) is examined in this thesis. The idea behind this concept is that given a graph g, the descriptors are calculated on g, and then used to create a feature vector that represents g. By computing all these descriptors for g, it is explicitly mapped to the real coordinate space, where the standard models for classification can be applied.

It is essential to understand the benefits of this method and the reasons for its further examination. As mentioned above, various approaches to classifying graphs exist. However, most graph analytics methods have a high computation and space cost, which is a problem that graph embedding seeks to solve. Graph embedding is an efficient way of reducing the increased complexity which is coupled with using graphs. Additionally, it maximizes the preservation of structural information and graph properties [9].

The goal of this bachelor thesis is to classify chemical compounds through graph embedding using the TDE. We select the three classifiers k-NN, SVM, and ANN for this task. Then the accuracies are compared to those achieved by reference systems. We hypothesize that the TDE can compete with modern classification methods while being computationally faster. If the hypothesis is verified, this would further strengthen the main advantage of graph embeddings, their low computational complexity. This approach could therefore be an alternative for classifying graphs when less computational power is of vital importance. A further question we seek to answer is which descriptors (i.e. features) are most important and consequently, a subgoal is finding the best features for classification. This procedure is commonly referred to as *feature selection* (FS), of which numerous variants exist. Feature selection is performed with the goals of reducing computation time, improving prediction performance, and achieving a better understanding of the data [10]. In light of the advantage of the computational efficiency of the TDE, another subgoal of this thesis is to compare the computation times to those of the reference systems.

The next few chapters form the main part of the thesis and are organized as follows. In Chapter 2, the theoretical concepts needed to understand the thesis are explained. Chapter 3 presents the current research and a similar study in the field. Chapter 4 provides insights into the methodology employed in this thesis, including topics like the embedding process, the data sets, and the reference system. The results are presented and analyzed in Chapter 5, before the thesis concludes with Chapter 6, summarizing the findings and suggesting future research directions.

Chapter 2

Basic Concepts

In this chapter, the theoretical basics needed to understand the thesis are covered. The concepts are divided into their own sections. This chapter begins with the basic concepts of graph theory in Section 2.1, then moves on to introduce different descriptors in Section 2.2, and finally, Section 2.3 gives a brief overview of the classifiers we use.

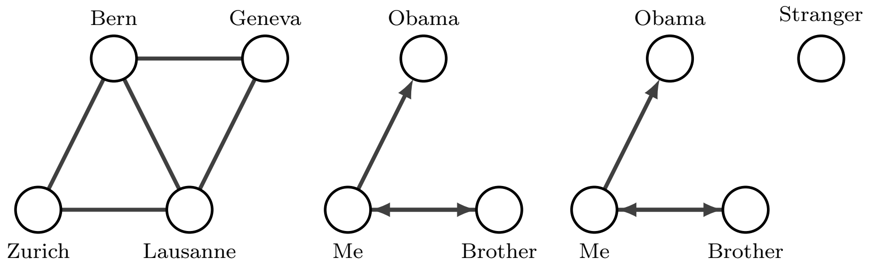
2.1 Graph Theory

A *graph* is a mathematical concept. One reason for using graphs is the representational power they possess. These concepts are explained to ensure the comprehension of the different definitions of the descriptors in Subsection 2.2. We also introduce some notations which will be used in Section 2.2 for defining the descriptors. First, we begin with the definition of a graph.

Definition 2.1 (Graph [11]**).** Let L_V and L_E be finite or infinite label sets for nodes and edges, respectively. A graph g is a four-tuple $g = (V, E, \mu, \nu)$, where

- V is the finite set of nodes,
- $E \subseteq V \times V$ is the set of edges,
- $\mu :\to L_V$ is the node labeling function, and
- $\nu :\to L_E$ is the edge labeling function.

For example, in the case of a molecular graph (a graph that represents a molecule), the labels of the nodes would represent atoms and the edges the bonds between the atoms. Then for a molecular graph g which contains the node v in its set V, $\mu(v) = H$ tells us that the node v is a hydrogen atom.



(a) Undirected Graph (b) Directed Graph (c) Not Connected Graph

Figure 2.1: Three examples for graphs showing the difference between directed/connected and undirected/not connected graphs.

A distinction can be made between *undirected* and *directed* graphs. An edge $e \in E$ is represented as a pair of nodes (u, v), where $u \in V$ represents the source node and $v \in V$ the target node for a directed edge, i.e., the edge is pointing from u to v. Undirected graphs can be modeled by additionally inserting the reverse edge (v, u) for every edge e into E. They receive the same label and so consequently $(u, v) = (v, u)$. Since every edge goes both ways, the direction can be ignored [11]. The difference between directed and undirected graphs is visualized in Figure 2.1. The first Graph 2.1(a) is an undirected graph that represents roads that connect Swiss cities. Since every road connects two cities in both ways we use undirected edges. Graph 2.1(b) is directed and represents whether one person knows another. For example, I know who Barack Obama is but he has never heard of me. Therefore, this is represented as a one-sided connection (i.e. a directed edge) from myself to Obama. Consider a scenario where my brother knows who I am and I know him but he has never heard of Obama. We employ two directed edges representing the connection between my brother and me, and not a single edge between my brother and Obama since neither of them know each other.

The next definition is that of *neighbors*. If two nodes are connected to each other with an edge, they are called neighbors or *adjacent* to each other. This information can also be represented in the format of a matrix, known as the *adjacency matrix*.

Definition 2.2 (Adjacency Matrix [11]**).** Let $g = (V, E, \mu, \nu)$ be a graph with $|g| = n$. The adjacency matrix $\mathbf{A} = (a_{ij})_{n\times n}$ of a graph g is defined by

$$a_{ij} = \begin{cases} 1 & \text{if}(v_i, v_j) \in E \\ 0 & \text{otherwise} \end{cases},$$

where v_i and v_j are nodes from g, i.e., $v_i \in V$.

The next important definition is the *degree* of a node v, which is the number of neighbors which that node has [12]. It will be referred to as $deg(v)$. For example, in Graph 2.1(a) $deg(Zurich) = 2$ and $deg(Bern) = 3$.

A *path* is a sequence of nodes and edges in which nodes cannot appear more than once. A path between u and v exists if there are nodes and corresponding edges that form a connection between u and v. For example, in Graph 2.1(a) a path from Zurich to Geneva exists. In fact, multiple different paths exist. Another example of a path would be the one going from Zurich, to Bern, to Lausanne, to Geneva. The *shortest path* for nodes u between v is the path with the smallest number of edges necessary to form a connection between u and v. For nodes Zurich and Geneva, one of the shortest paths could be Zurich, to Bern, to Geneva. The *distance* between two nodes u and v in an undirected graph is the number of edges in the shortest path and will be written as $d(u, v)$ [12].

So far we have only presented examples of *connected graphs*. Examining Graph 2.1(b), it is observable that paths exist between all three nodes. This means that these three nodes are *connected* to each other. Say we want to represent a stranger who is mutually unknown to the three current people. To do this, a fourth node with no edges connecting it to any of the first three nodes is added in Graph 2.1(c). We assign the label "Stranger" to it. Although now a node exists that is not connected to any of the other nodes, this still represents a single graph and not two different ones. The Graph 2.1(c) containing all four nodes is now considered to be *not connected graph*. Conversely, a graph is called a connected graph, if, for an arbitrary node, there is a path to every other node [12]. Some of the descriptors in Section 2.2 only make sense when considering connected graphs. Therefore, we only use connected graphs for this thesis, and those that are not, are filtered out in the preprocessing of the data sets.

2.2 Topological Descriptors

In this section, all the descriptors used for the graph embedding are listed and defined. The choice of these descriptors is not contingent on any specific criteria other than being *topological descriptors* (with the exception of the last three which make use of edge or node labels). Hayat et al. [13] define a topological descriptor as follows: "A topological descriptor/index (...) is a numerical value associated with chemical constitution for correlation of chemical structure/network with various physical properties, chemical reactivity or biological activity." The focus is not on the interpretation or the real-world applications, and hence, such aspects are not detailed in the list. For the following descriptors, we consider a graph g, where $g = (V, E, \mu, \nu)$ as defined in Definition 2.1.

Nodes and Edges The first two descriptors are simply the number of nodes and the number of edges of a graph g. Formally, the first descriptor is defined as

$$No(g) = |V|.$$

The second one is formally defined as

$$E(g) = |E|.$$

First Zagreb Index The *first Zagreb index* was first introduced in 1972 [14] and for a graph g, is the sum of all squared node degrees. Formally, it is defined as

$$Z_1(g) = \sum_{v \in V} deg(v)^2.$$

Narumi-Katayama Index The *Narumi-Katayama index* (or *simple topological index*) was defined by Narumi and Katayama in 1984 [15]. The Narumi-Katayama index of a graph g is the product of all node degrees. Formally, it is defined as

$$NK(g) = \prod_{v \in V} deg(v).$$

Polarity Number The *polarity number* was defined by Wiener in 1948 [16]. The polarity number of a graph g is defined as the number of unordered pairs of vertices $u, v \in V$ that are at a distance of 3 from each other. Formally, it is defined as

$$P(g) = \sum_{v \in V} \sigma(d(u, v)),$$

where

$$\sigma(x) = \begin{cases} 1 & \text{if } x = 3 \\ 0 & \text{otherwise} \end{cases}.$$

Wiener Index The *Wiener index* was introduced in 1947 by Wiener [17]. The Wiener index of a graph g is defined as the sum of the lengths of the shortest paths (i.e. distances) between all pairs of nodes. Formally, it is defined as

$$W(g) = \sum_{u \in V} \sum_{v \in V} d(u, v).$$

Randić Index The *Randić index* (or *connectivity index*) was introduced in 1975 by Randić [18]. For a graph g, it is formally defined as

$$R(g) = \sum_{e \in E} \frac{1}{\sqrt{deg(u) \cdot deg(v)}},$$

where $e = (u, v)$.

Estrada Index The *Estrada index* was introduced by Estrada in 2000 [19]. For a graph g, it is formally defined as

$$EE(g) = \sum_{i=1}^{n} e^{\lambda_i},$$

where λ_i are the eigenvalues of the adjacency matrix of g and $n = No(g)$. Since the goal is to embed the graph, we do not want a complex result, which may occur due to the eigenvalues. Therefore, if the Estrada index is a complex number, we estimate the Estrada index using an approximation [20]. The approximation is defined as

$$EE^{*}(g) = \frac{n \cdot (e^{k} - e^{-k})}{2 \cdot k},$$

where $n = No(g)$, $m = E(g)$, and $k = \sqrt{\frac{6m}{n}}$.

Balaban-J Index The *Balaban-J index* was proposed by Balaban in 1982 [21]. For a graph g, it is formally defined as

$$B(g) = \frac{q}{\mu + 1} \sum_{i,j}^{n} \frac{1}{\sqrt{s_i \cdot s_j}},$$

where $n = No(g)$, $q = E(g)$ and $\mu = q - n + 1$. s_i is defined as the sum of the distances from node i to all other nodes. Formally, $s_i = \sum_{v \in V} d(i, v)$.

Szeged Index The *Szeged index* was defined by Gutman et al. in 1995 [22]. For a graph g, it is formally defined as

$$Sz(g) = \sum_{e \in E} n_v(e|g) \cdot n_u(e|g),$$

where $e = (u, v)$ and $n_v(e|g)$ is the number of vertices lying closer to u than v and respectively $n_u(e|g)$ is the number of vertices lying closer to u than v.

Padmakar-Ivan Index The *Padmakar-Ivan index* was defined in by Khadikar in 2000 [23] and for a graph g, is formally defined as

$$PI(g) = \sum_{e \in E} (n_{eu}(e|g) + n_{ev}(e|g)),$$

where $n_{eu}(e|g)$ is the number of edges lying closer to vertex u than v and respectively $n_{ev}(e|g)$ is the number of edges lying closer to vertex v than u. Note the key difference between the Padmakar-Ivan and the Szeged indices lies in counting edges (rather than vertices) that lie closer to a vertex.

Molecular Topological Index The *Schultz molecular topological index (MTI)* was proposed by Schultz in 1989 [24]. The MTI of a graph g is formally defined as

$$MTI(g) = \sum_{i=1}^{n} \sum_{j=1}^{n} deg(i) \cdot (a_{ij} + d(i,j)),$$

where $n = No(g)$, $deg(i)$ is the degree of vertex i in g, and a_{ij} is the (i,j)-th entry of the adjacency matrix A of g.

Modified Zagreb Index The *modified Zagreb index* was proposed by Manzoor et al. in 2020 [25]. The modified Zagreb index of a graph g is formally defined as

$$ReZG_1(g) = \sum_{e \in E} \frac{deg(u) + deg(v)}{deg(u) \cdot deg(v)},$$

where $e = (u,v)$.

Hyper Wiener Index The *hyper Wiener index* was proposed in 2012 by Khalifeh et al. [26]. The hyper Wiener index of a graph g is formally defined as

$$WW(g) = \frac{1}{2} \sum_{u \in V} \sum_{v \in V} (d(u,v)^2 + d(u,v)).$$

Neighborhood Impurity The *neighborhood impurity* is calculated following the methodology used by Li et al. [27]. To define neighborhood impurity for a graph, the neighborhood impurity (ni) for a node v needs to be defined first.

$$ni(v) = |\mu(u) : u \in N(v), \mu(u) \neq \mu(v)|,$$

where $\mu(v)$ is the node label of v and $N(v)$ is the set of nodes which are adjacent to node v. Formally, for a graph g, the neighborhood impurity can now be defined as

$$NI(g) = \frac{1}{k}\sum_{v \in V} ni(v) \cdot \sigma(ni(v)),$$

where $ni(v)$ is the neighborhood impurity for a node,

$$\sigma(x) = \begin{cases} 1 & \text{if } x > 0 \\ 0 & \text{otherwise} \end{cases},$$

and $k = \sum_{v \in V} \sigma(ni(v))$.

Label Entropy The *label entropy* is calculated following the methodology used by Li et al. [27]. Formally, for a graph g, the label entropy is defined as

$$H(g) = -\sum_{i=1}^{q} p(l_i) \cdot log_2(p(l_i)),$$

where g has q different labels: $l_1, l_2, ..., l_q$, and $p(l_i)$ is the probability of a node having label l_i.

Average Edge Weight The graphs in the data sets used for this thesis all have edge labels, where the edges are numerically encoded for the types of edges that they represent (e.g. single, double, or tripple bonds). The *average edge weight (AEW)* is calculated as

$$AEW(g) = \frac{1}{m}\sum_{e \in E} \nu(e),$$

where $m = E(g)$, $e = (u, v)$, and $\nu(e)$ is the edge label of e.

2.3 Classifiers

Once the graph data sets are embedded, one can perform classification. Classification is performed by training the classifiers on training data and then using the trained classifier to perform predictions on the test data. As mentioned in Chapter 1, there are many different classifiers that can achieve this. For this thesis, three different classifiers are chosen: the k-NN model [28], the SVM [29], and the ANN [30]. Since the classifiers are only tools to compare the classification methods they are not the focus of the thesis. Therefore they are only presented briefly. For more detailed information concerning the classifiers, refer to the work by Rjini et al. [31]. Still, we summarize the key aspects that they mention in the list below. When using classifiers, we typically follow two stages: the training stage and the testing stage. In the training stage, we use a batch of data, called X_train, and its corresponding labels, y_train, to train the model. Following that, in the testing stage, we present the test data, referred to as X_test, and assign a label to each sample s in X_test.

k-NN The k-NN classifier is one of the simplest options and belongs to the supervised learning models. This means that the training data is correctly labeled when training the model. It is said to achieve good results if the optimal k is found. The labels are assigned to sample s in X_test by taking the majority of the labels of the k nearest neighbors (the other samples in X_train) to s.

SVM The SVM is also a supervised learning classifier and achieves good accuracies in various domains. It learns an optimal hyper-plane between the two classes in the training step and then assigns the labels to the samples in X_test based on which side of the hyper-plane they lie on.

ANN This classifier is a mathematical model. The following description is very simplified, but essentially there are a number of elements that are organized into layers and connected to each other. The connections are then tuned during the learning process using backpropagation. The trained ANN returns predictions of how high the probabilities of the sample belonging to each label are. The label with the highest probability is then chosen as the assigned label.

Chapter 3

Current Research

As mentioned in Chapter 1, we hypothesize that a TDE is an efficient classification method and achieves competitive results despite it being a very simple approach. This chapter covers relevant research and motivates our reason for posing this hypothesis. The following section covers some alternative graph embeddings and graph kernels in order to give an understanding of which other graph-based PR methods exist (apart from the TDE method). In the second Section 3.2, we present the paper of Li et al. [27] and their success with global topological predictors.

3.1 Graph Embeddings and Graph Kernels

This section is based on the review by Bunke et al. from 2012 [8] which covers graph kernels and graph embeddings. In the review, they additionally show that graph embeddings or graph kernels are potentially useful alternatives when applied to graph-based document analysis. However, the objective here is to shed light on other ways to perform graph-based PR, so that aspect of the paper is not presented.

The mathematical details can be found in the paper and are not presented here, rather the key aspects of graph kernels are presented to give a rough understanding of this alternative. The basic idea behind graph kernels is that they compute the similarity between pairs of graphs based on some common patterns that they share. Graph kernels are used due to their capability of coping with non-linear data and complex data structures like graphs. It is possible to compute the geometrical properties of graphs in an implicitly existing feature space without having to compute the mapping to it. This allows graph kernels to cope with the lack of mathematical structure in the domain of graphs. Two examples of graph kernels are presented in the following list.

Convolution Kernels To calculate the similarity between a pair of graphs, the

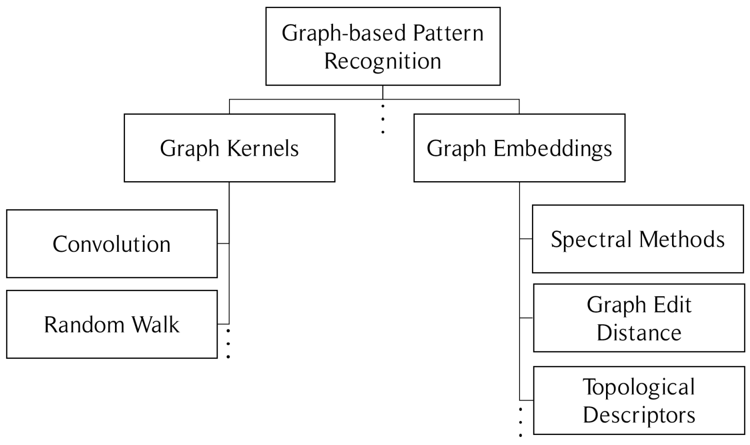


Figure 3.1: An overview of different graph-based PR methods, divided into graph kernels and graph embeddings.

convolution kernels first decompose the graphs into smaller parts. This is to simplify the graphs. The similarity product is then calculated for all the smaller parts and finally, the products are all summed up.

Random Walk Kernels These kernels make use of Markov chains which are used to define random walks. The mathematical background for random walks can be found in the paper by Lovász [32]. The idea is that for two graphs, the number of random walks which contain all or some of the same labels are used to measure the similarity between them.

The alternative to graph kernels is graph embeddings. Similarly to graph kernels, the goal of embedding graphs is to simplify graphs in order to apply more efficient algorithms, while retaining the structural properties of graphs. However, unlike graph kernels, a graph embedding explicitly maps the graph into the real vector space $\mathbb{R}^n$, effectively transforming a graph into a vector. This allows for the use of standard algorithms which require vectorial data. Some examples of graph embeddings would be:

Spectral Methods This approach uses the eigenvalues of the adjacency or Laplacian matrix in order to try and characterize the structural properties of graphs. However, this approach is limited in its flexibility and robustness since it is not able to cope with graphs with extensive label alphabets or data sets with a large amount of noise.

Graph Edit Distance (GED) For this approach, the similarity of two graphs is measured by calculating how expensive it is to turn the first graph into the second one. Graph operations like deletions, insertions, and substitutions are given a price. The more of these operations that need to be performed to make the graphs isomorphic (i.e. equal), the higher the GED of that graph pair is. It has the advantage of being applicable to many different graphs (directed and undirected, with and without node/edge labels) and is a flexible, robust method of embedding graphs. However, its major drawback is the computational complexity needed to calculate the GED between two graphs.

Topological Descriptors As mentioned in Chapter 1, the TDE is a type of embedding. The details are presented in Chapter 4, Section 4.1.

3.2 Similar Research

In their paper, Li et al. [27] present the classification of graphs using global topological and label attributes with an SVM. Using 20 different descriptors, graph embedding is performed on a total of 19 different data sets. Four of the descriptors used by Li et al. overlap with the set of descriptors selected in this thesis (number of nodes/edges, neighborhood impurity, label entropy). The data sets contain chemical compounds (i.e. molecules), proteins, and cell graphs. In total, seven data sets containing chemical compounds are chosen and we will only focus on these seven data sets here. After performing the embedding Li et al. normalize their data sets and perform the classification. Their approach using the topological descriptors will be referred to as graph feature embedding (GF). They compare GF to numerous kernels (fast geometric random walk, shortest path, Weisfeiler-Lehman subtree (WL), graphlet, Ramon-Gärtner subtree, CORK).

Li et al. achieve statistically significant accuracy improvements on the MUTAG, and PTC_MR data sets by applying GF with a z-normalization. Additionally, they achieve the best accuracy when using GF combined with a range normalization on the PTC_FR data set. However, this accuracy is not significantly better. On the other four chemical data sets the WL kernel performs significantly better. They believe it may be due to the tree-like data sets. Since the WL kernel is a subtree kernel, it is well suited for those data sets. Although the WL kernel outperforms the GF for those approaches, it does so at the expense of significantly longer computation times.

For the chemical compounds, Li et al. show that GF is one to three orders of magnitude faster on every data set when compared to the other kernels. Regarding the training time, they show that only CORK manages to compete with GF. By performing a scalability study they show that the runtime for the GF topological features extraction is linear to the number of graph instances.

Li et al. also perform the classification after omitting the labels and show that it barely has an impact on the GF approach. They believe that GF may therefore rely mainly on topological features. In contrast, WL suffers from performing classification without labels. To find out how important each feature is, an SVM-wrapper FS method is performed. The chosen features are found to depend on the normalization and data sets and that simple topological features are typically enough to retain the structural information and properties of graphs.

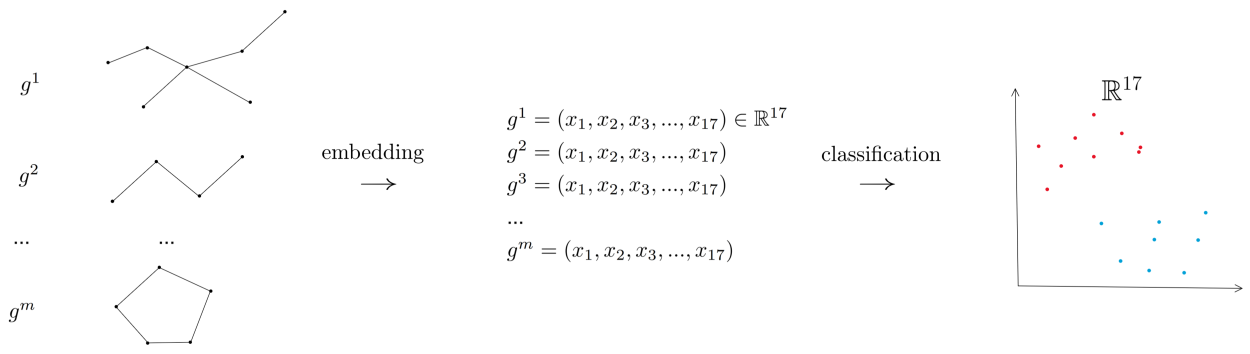
Chapter 4

Embedding by Topological Descriptors

In this section the details of the employed method are presented and explained. First, the process of embedding a graph data set is explained, second, the various data sets are introduced and third, the steps used for the classification are addressed. Finally, the reference systems used for results comparison and the technical framework are presented.

4.1 Embedding a Graph

Embedding a graph describes the action of taking a graph and mapping it to a new n-dimensional feature space. In this thesis, this is done by calculating 14 different topological and three different label descriptors (so $n = 17$) for each graph. These descriptors are then used to create a feature vector for the graph. Each descriptor is represented in a different entry of the vector. By doing this, a graph is effectively transformed into a feature vector. This means that if given a graph g, the graph can be represented as a single point in the 17-dimensional space $\mathbb{R}^{17}$. From here on out g_1 is the notation we will use for the value resulting from calculating the first descriptor on graph g and g^m will refer to the m-th graph of a data set. After performing the embedding, every graph has the same descriptor in the respective dimension. This means that given graphs g^i and g^j with $i \neq j$, ${g^i}_1$ and ${g^j}_1$ store the result of the first descriptor calculated on graphs i and j, respectively. This allows us to compare the graphs after the embedding. A sketch is provided to visualize the process in Figure 4.1. The times it takes to compute these embeddings are saved so they can be analyzed.



g^1
g^2
...
g^m
...
embedding
$\longrightarrow$
$g^1 = (x_1, x_2, x_3, ..., x_{17}) \in \mathbb{R}^{17}$
$g^2 = (x_1, x_2, x_3, ..., x_{17})$
$g^3 = (x_1, x_2, x_3, ..., x_{17})$
...
$g^m = (x_1, x_2, x_3, ..., x_{17})$
classification
$\longrightarrow$
$\mathbb{R}^{17}$

Figure 4.1: This sketch illustrates the process of embedding and classifying a graph data set. On the left, a data set with m graphs is given. The first arrow represents the process of embedding each graph by calculating the 17 different descriptors. The middle part represents the newly embedded data set which now contains m different feature vectors, each representing a graph. The second arrow represents the classification step where a class is assigned to each sample. Two different classes are visualized using red and blue.

4.2 Data Sets

In total, eight different graph data sets have been chosen from the collection of the TUdata sets [33]. The chosen data sets all contain chemical compounds that are stored as graphs. The eight different data sets are: PTC_MR, PTC_MM, PTC_FM, PTC_FR, MUTAG, Mutagenicity, ER_MD, and DHFR_MD. The compounds in the PTC data sets are labeled according to the carcinogenic effect they have on rodents. They are divided into male mice (MM), male rats (MR), female mice (FM), and female rats (FR) [34]. The MUTAG data set contains aromatic and heteroaromatic nitro compounds that are labeled based on their mutagenic effect on the bacterium Salmonella typhimurium [35]. The samples of the Mutagenicity data set are encoded on whether the molecular structures are mutagenic or not [36]. In the context of this data set, mutagenicity is a property of a chemical compound that reduces its potential of becoming a marketable drug [37]. The chemical compounds in the MD data set are activities. They are labeled based on which spline descriptor an activity is assigned to. The DHFR data set contains dihydrofolate reductase inhibitors and the ER data set contains estrogen receptor ligands [38].

To give a more technical overview, Table 4.1 shows some basic descriptors on the data sets. Some key observations can be made from the table. The MUTAG data set is the smallest and contains small graphs. The PTC data sets also contain small graphs. Mutagenicity is the biggest data set and contains bigger graphs than the previously mentioned data sets, it has a very similar average degree value though.

The two MD data sets are the most complex ones as they have the highest number of nodes and edges. They hold a roughly 10 times higher average degree than the other data sets.

Attributes	**Size**	**∅ Nodes**	**∅ Edges**	**∅ Node Degrees**	**Max. Nodes**	**Max. Edges**
MUTAG	188	17.93	19.79	2.19	28	33
Mutagenicity	4046[1]	30.32	30.77	2.04	417	112
ER_MD	446	21.33	234.85	22.87	39	741
DHFR_MD	393	23.87	283.02	20.33	43	903
PTC_MR	344	14.29	14.69	1.98	64	71
PTC_MM	336	13.97	14.32	1.97	64	71
PTC_FM	349	14.11	14.48	1.98	64	71
PTC_FR	351	14.56	15.00	1.99	64	71

Table 4.1: This table contains some basic descriptors calculated on the data sets. The first column contains the number of graphs per data set. The ∅ denotes the average for the number of nodes, edges, and average node degrees that the graphs in the respective data set have. The max. nodes or edges denote the highest number of nodes or edges that a single graph has in the data set.

4.3 Methodological Steps for Classification

Preliminary data preprocessing is carried out prior to classification. We perform a z-normalization on the data sets. That means that every sample is transformed by subtracting the mean and dividing it by the standard deviation. For each feature, this centering and scaling process is executed independently due to the significant differences in variances across features. For instance, the Narumi-Katayama index values on the ER_MD dataset range from 81 to multiple trillions, whereas the label entropy ranges from 0 to 1.7. The z-normalization performs the best (compared to the range normalization or none at all) in the paper of Li et al. [27].

Two FS algorithms are applied in this thesis, the first one being the *minimum redundancy - maximum relevance selection* (mRMR) [39], and the second one is the *sequential forward selection* (SFS) [40]. The mRMR selection is independent of the classifier. The goal of the mRMR selection is to only select the minimal-optimal features and therefore, we choose to only pick five features per data set. Each new feature is chosen by considering the relevance it has with the labels and the redundancy of the already-picked features with the new one. These scores per feature are calculated on the data set before the train and test splits are created.

[1]The Mutagenicity data set originally contains 4337 graphs, however, 291 of them are not connected so they are filtered out. This is because some descriptors (e.g. the Wiener index) result in infinity if the graphs are not connected.

Classifier	Parameter	Grid
k-NN	metric	euclidean, manhattan, cosine
	k	1 - 31
SVM	C	0.001, 0.01, 0.1, 1, 10, 100
	γ	0.001, 0.01, 0.1, 1, 10, 100
	kernel	radial basis function, sigmoid

Table 4.2: This table contains the different hyperparameters per classifier on which the gird searches are performed.

SFS on the other hand needs a classifier (k-NN, SVM, or ANN) to select features and therefore the features are selected separately for each classifier. SFS selects features in a greedy fashion. It starts with an empty feature set and one by one adds features by assessing the score of the previously selected features combined with a new feature. The score of a potential new set is attained using cross-validation (CV) on the training set[2] and the set that scores the highest is then selected to continue. This iterative process continues until 10 features are selected. Pseudocode for mRMR and SFS algorithms is provided in Appendix A. The selected features and their scores are saved, so they can be analyzed once the classification is finished.

Before we perform the FS, the data sets are divided into train and test splits. The train splits consist of 80% of the samples. The test splits are not touched until the final predictions are made. These predictions are used to measure the achieved accuracies of a data set and classifier.

Next, for the k-NN and SVM models, a grid search is applied to find the optimal hyperparameters per data set. For this, a 10-fold CV is used. The grids for the different hyperparameters are presented in Table 4.2. For more detail on the individual hyperparameters, the scikit-learn documentation can be consulted [41]. No hyperparameter selection is performed on the ANN. Instead, its hyperparameters are chosen based on intuition and set as follows: input dimension (ID): number of selected features, output dimension (OD): 2, hidden layers: average of ID and OD, epochs: 300, loss: cross-entropy, batch-size: 32, optimizer: stochastic gradient descent, scheduler: cyclic learning rate policy.

Once training is finished for the different classifiers, the labels are predicted and the accuracies are reported. They are then compared to the accuracies that the respective reference system achieves and the significance is calculated using the Z-test. The results are presented in Chapter 5.

[2]For the k-NN and SVM classifiers, a five-fold CV is used. For the ANN a three-fold CV is used in order to reduce computation time.

4.4 Reference Systems

The reference systems are used to calculate accuracies, against which the accuracies from the TDE methods are compared. Three reference systems are used in this thesis, one for each classifier. Each reference system is designed to use a classifier that is the same as or similar to its TDE counterpart to ensure comparable results. For the k-NN and SVM models, the reference system employs the GED[3]. We use the GED because it is commonly used in practice. For the k-NN reference, we compute a distance matrix that contains the distances for all pairs of graphs and feed it into a k-NN model. The classification process is very similar to the TDE k-NN. Instead of calculating the distances using a metric (e.g. Euclidean), the precalculated distance matrix now supplies the distances. For the SVM we calculate a similarity kernel (a negative distance matrix) and supply this to an SVM classifier. More detail on how the GED is incorporated into kernel functions can be found in the book by Neuhaus et al. [42]. After computing the kernels, the hyperparameter selection is performed. The same hyperparameter grid which is used for the TDE is supplied with the exception of the metric and algorithm. These are replaced by an additional α parameter, which is used for the computation of the GED. It ranges from 0.05 to 0.85 with steps in between increasing by 0.2. The third reference system, employed for comparing the accuracies of the ANN, is a graph convolutional network (GCN) [43]. This is used because it is another common way of classifying graphs and because the ANN and GCN are both deep learning models. We use a simple GCN and do not perform a hyperparameter selection on it. We use the same hyperparameters that the ANN uses.

Once the final accuracies are computed, we perform statistical tests to find out whether our results are statistically significant. An accuracy difference is deemed statistically significant (either superior or inferior) if the computed Z-test score falls below the specified significance level ($\alpha = 0.05$). It is assumed that the two methods of calculating the accuracies are equally as good. The Z-test then tells us what the probability of getting the compared accuracies is, given this assumption. If this probability is sufficiently small, the difference in accuracies is statistically significant. For example, assume the Z-test is 0.03 when comparing a TED method to the reference system. This would mean that there is a 97% chance that the two methods being compared are not the same, therefore we say that the difference in accuracy is statistically significant. Various times are logged and saved for the reference systems to later be compared to those of the TDE approach.

[3]Note that this is not a valid GED embedding as mentioned in Section 3.1.

4.5 Technical Framework

The implementation of the code of the thesis relies on the following technical components. The selected coding language is Python [44]. Amongst others, the main libraries chosen are NumPy [45], pandas [46], scikit-learn [41], PyTorch [47], and PyTorch Geometric (PyG). The following list presents the main purposes of the libraries for the thesis:

NumPy It makes it possible to compute descriptors more efficiently, particularly when it comes to executing matrix calculations.

Pandas It is used for data manipulation and for storing and reading the embeddings in comma-separated values (CSV) files.

scikit-learn It provides the SVM and k-NN models, as well as numerous other methods used for the classification. For example, train-test split or grid search methods are available.

PyTorch It provides the ANN model and related methods and objects.

PyG It is a library built on PyTorch and provides access to the graph data sets, as well as the GCN model and graph-related methods.

Chapter 5

Experimental Evaluation

This chapter lays out the outcomes of our experimental work. The focus lies on the comparison of accuracies, so they are presented first in Section 5.1.1. Subsequently, we present the results of our feature study in Section 5.1.2 and the recorded computation times of our experiments in Section 5.1.3. The critical discussion of the presented results follows in Section 5.2. The hyperparameters which are chosen during the classification process are not presented but can be found in Appendix B. We choose not to present them explicitly because they are technical details and would need further explanation, while potentially not being crucial to our discoveries. Rather we focus on presenting the key findings.

5.1 Results

5.1.1 Classification

In Table 5.1 the achieved accuracies for all data sets are displayed. The accuracies are grouped per classifier and per data set. The results for the different FS methods (SFS, mRMR, no FS) compared to the respective reference system are visible. No FS means that no FS is applied to the data set. The accuracy of the methods which score the highest is marked in bold. Statistically significant improvements over the reference system are marked with ① and statistically significant deteriorations are marked with ❶. Five runs are performed for both the ANN and GCN. The average accuracy is deemed significant if the majority of these runs are statistically significant. Additionally, for the ANN, we display the standard deviations over the five runs.

For the MUTAG data set our approach achieves a statistically superior accuracy when applying the k-NN classifier combined with either SFS or without any FS. Although not statistically significant, the SVM classifier shows the same pattern that

		MUTAG	Mutagenicity	ER_MD	DHFR_MD	PTC_MR	PTC_MM	PTC_FM	PTC_FR
k-NN	**Reference**	86.84	**72.28**	75.56	**77.22**	**63.77**	63.24	**60.00**	**59.15**
	SFS	**97.37** ①	71.29	**83.33** ①	70.89	53.62	58.82	**60.00**	54.93
	mRMR	89.47	70.42	74.44	70.89	55.07	57.35	58.57	52.11
	No FS	**97.37** ①	70.30	82.22	**77.22**	62.32	**64.71**	51.43	54.93
SVM	**Reference**	89.47	66.46	68.89	68.35	56.52	64.71	60.00	56.34
	SFS	**94.74**	72.65	**81.11** ①	**77.22**	**63.77**	60.29	**61.43**	54.93
	mRMR	89.47	72.65	75.56	70.89	62.32	66.18	60.00	56.34
	No FS	**94.74**	**73.27**	80.00 ①	75.95 ①	**63.77**	**72.00** ①	**61.43**	**57.75**
ANN	**Reference**	82.63 ± 1.44	**76.14 ± 0.46**	**61.78 ± 1.27**	**70.38 ± 3.05**	61.16 ± 1.21	**76.76 ± 1.61**	**60.29 ± 3.26**	**60.85 ± 1.18**
	SFS	72.63 ± 4.78	61.23 ± 0.56	58.89 ± 0.00	68.60 ± 0.57	60.29 ± 0.79	64.71 ± 0.00 ❶	59.71 ± 0.64	56.34 ± 0.00 ❶
	mRMR	71.58 ± 3.87 ❶	63.96 ± 0.42	60.22 ± 1.09	68.60 ± 0.51	59.71 ± 1.69	64.71 ± 0.00 ❶	59.71 ± 0.64	58.29 ± 2.28 ❶
	No FS	**83.68 ± 4.32**	63.34 ± 0.29	57.34 ± 2.02	68.86 ± 0.70	**64.06 ± 2.38**	64.71 ± 0.00 ❶	59.71± 0.64	56.34 ± 0.00 ❶

Table 5.1: This table presents the achieved accuracies (in %) of the eight data sets depending on which methods and classifiers are applied. The three FS methods of the TDE (SFS, mRMR, no FS) are compared to the reference system. ①/❶ marks a statistically significantly superior/inferior result compared to the reference system (Z-test using $\alpha = 0.05$). The highest accuracies per data set and classifier are marked in bold.

the k-NN classifier does. For the ANN model, we achieve the best result when using the TDE without FS. The combination of the ANN with mRMR seems particularly weak, resulting in the only statistically lower accuracy of a TDE application on the MUTAG data set. We achieve further significant improvements when using the TDE on the ER_MD and DHFR_MD data sets with the SVM and k-NN models. Conversely, the reference system beats the ANN both times. The PTC data sets do not profit much from the application of the TDE overall. Over the four data sets and three TDE methods, only a single accuracy is significantly better than the reference system. As is the case with the MD data sets, the ANN does not pair well with the TDE on these data sets. In fact, we can observe a significant deterioration in half the PTC data sets. This combination consistently yields small standard deviations. For the PTC_MM and PTC_FM data sets the results are equal for all the TDE applications for the five ANN runs. This may be because for these data sets the ANN learns the weights and biases in a manner that ends up with the same results, no matter which method of TDE is applied. Although the reference system outperforms the TDE in combination with ANN, the TDE combined with an SVM scores higher than its reference system every time. Overall, our comparison of different methods reveals that applying the reference system and the TDE without FS outperform the others, achieving the best results in approximately half of all cases each (11/24 and 10/24 times). SFS does not lag far behind, performing the best eight times. Conversely, mRMR consistently underperforms, establishing it to be the least effective FS method.

5.1.2 Feature Study

During the classification we also track which features are selected in the FS process. Figure 5.1 illustrates how important a feature is per FS method. The higher the points, the more important we believe it to be. The points are calculated based on our ranking system that takes into account in which order the features are selected and if they appear in the selection at all. For SFS 10 features are selected in a sequential manner per data set. For each data set, the first feature is awarded 10 points, the second nine, and so on until the final selected feature is awarded one point. The other features which are not selected receive zero points. The same pattern is used for the mRMR where five points are awarded to the first feature and one point to the final one (since we only select five features for mRMR per data set). The points per feature are then summed up over all data sets to get the total points for a feature per FS method. When comparing the plots, note that SFS plots all have a y-axis with a maximum of 75 points, whereas the maximum for the mRMR plot is set to 25 points.

In the k-NN SFS plot, the polarity number and the Padmakar-Ivan index stand out, scoring high with 57 and 56 points, respectively. Other descriptors, including Narumi-Katayama, Balaban-J, and Estrada indices, follow closely. In the SVM SFS plot, we can see a very similar pattern to the k-NN plot. The top five features are the same, followed by the same descriptors in the exact order from rank six to 11. Consequently, the two classifiers also share the bottom four descriptors. The SVM SFS plot has the highest variances of the three SFS plots with the Balaban-J index taking the lead at 70 points, clearly outperforming the other features by 15 or more points. The only feature scoring zero points in all three SFS plots is the modified Zagreb index. In the ANN SFS plot, the distribution of points among features is relatively even. Similar to the SVM SFS plot, the Balaban-J index remains at the forefront with 53 points. The Narumi-Katayama index and polarity number are among the top five once more. For the mRMR plot we once again see the Balaban-J and Narumi-Katayama indices in the top five. Other than that, the mRMR differs substantially in the selected features when compared to the others.

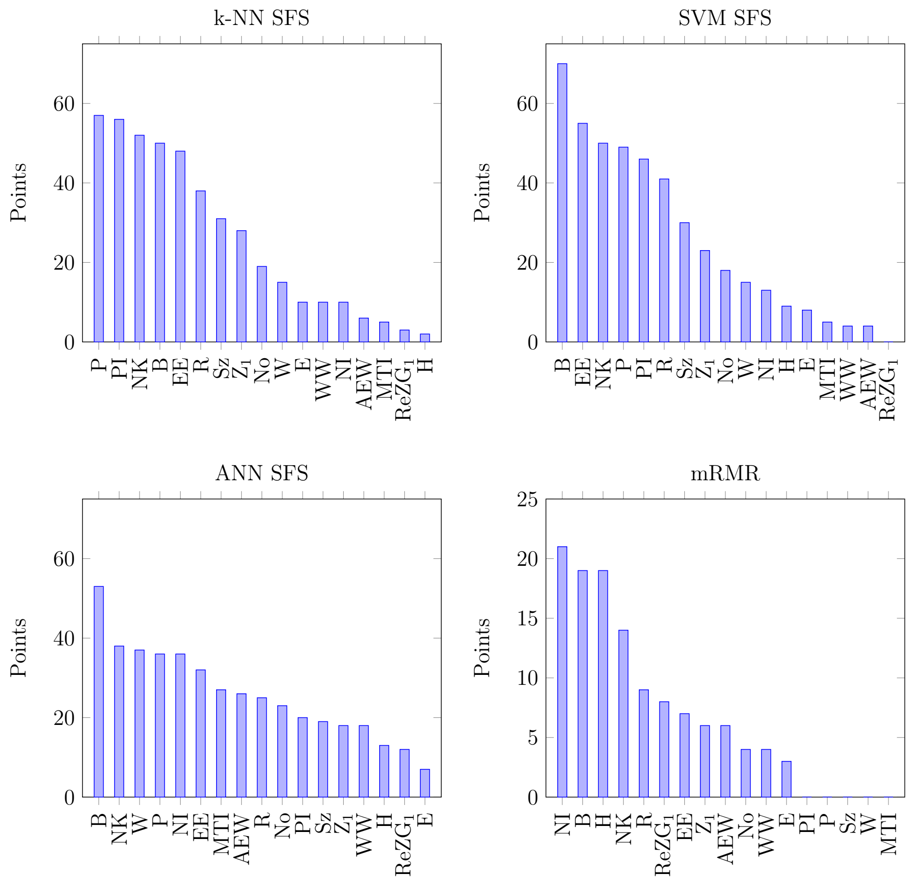


Figure 5.1: The bar plots show the number of points each feature is awarded by our ranking system. The higher the points, the more important a feature is. The features are grouped per classifier for the SFS and the points are summed over the eight data sets. The mRMR is also shown here but it has its own scale since only five features are selected in mRMR and therefore the distribution of points is slightly different. Since mRMR does not distinguish between classifiers only one plot is necessary to present the points for this method. Feature names are denoted by using their formula names from Chapter 2.2.

Data set	GED vs. TDE Time [h:m:s] Kernelization	Data Set Embedding
MUTAG	00:00:24	00:00:03
Mutagenicity	05:47:41	00:01:15
ER_MD	00:04:48	00:00:55
DHFR_MD	00:04:39	00:00:53
PTC_MR	00:01:08	00:00:04
PTC_MM	00:01:05	00:00:04
PTC_FM	00:00:53	00:00:04
PTC_FR	00:00:58	00:00:04

Table 5.2: This table presents the times it takes to calculate the GED kernels and compute the embeddings per data set.

	ANN Time [m:s]		
Data set	Reference	SFS	No FS
MUTAG	00:17	01:40	00:05
Mutagenicity	16:38	29:02	01:34
ER_MD	02:15	03:21	00:23
DHFR_MD	01:05	02:58	00:13
PTC_MR	00:33	03:01	00:11
PTC_FM	00:28	02:54	00:09
PTC_MM	00:27	02:35	00:14
PTC_FR	00:28	12:10	00:09

Table 5.3: This table presents the total time it takes to perform the classifications for the reference system, SFS and no FS.

5.1.3 Computation Times

The final two tables give us an impression of the computation times it takes when applying the different classification methods. Table 5.2 presents the times it takes to compute the GED kernels and the embeddings per data set[1]. For each data set, the GED kernelization time is a multiple of the time it takes for the descriptor computation. With Mutagenicity being the largest data set, it exhibits the most notable difference with the factor being about 278. For the other data sets, kernelization takes five to 17 times longer than the embedding computation. Computing the embedding is substantially faster than computing the kernels needed for the GED alternative for all data sets.

[1]The times presented in this table are to be seen as approximations only since the computational capability of the high-performance computing cluster is not something we can control perfectly. However, although the times may vary slightly from run to run, they are generally consistent.

Table 5.3 shows the different computational times it takes to perform the classification of the data sets with either the ANN or GCN. One can see that the ANN without FS is the fastest approach. However, if FS is applied, then the GCN ends up being faster than the ANN. Note that the embedding time for the two TDE approaches is not included in this table, since the embedding is done beforehand. So if the complete approaches are to be compared, the respective seconds from Table 5.2 would need to be added to the times in the table.

5.2 Discussion

Due to the observations made in Section 5.1 we believe that the TDE proves itself to be a potential alternative to other modern graph classification approaches when combined with a k-NN or SVM classifier and either SFS or no FS is performed. It achieves statistically equal or better results and manages to do so in significantly less time due to its computational simplicity. The TDE seems to perform exceptionally well when combined with an SVM. Given the comparisons to the reference system, it seems that the TDE coupled with an ANN does not yield optimal results, as the GCN outperforms it in six out of eight instances. We generally observe statistically significant deteriorations compared to the reference systems when a TDE method combined with an ANN is applied. Not once does the mRMR emerge as the top method, which suggests that this FS technique is not suitable for the TDE. Since with SFS and no FS we use 10 and 17 features, we believe the poor performance of mRMR may be caused by the lack of available features (only five are used for mRMR, resulting in a much smaller amount of data for classification). Considering the small size of most of the data sets, it is worth noting that the resulting accuracies might vary based on the chosen train-test split. This is due to our chosen method of using the test split at the very end and not applying CV to the entire data set. The splits may by chance be especially easy or hard to classify for the models.

It is worth noting that between the two FS methods mRMR and SFS, the method with the best accuracies is SFS combined with an SVM or k-NN classifier. ANN and SFS, or mRMR combined with any classifier of the classifiers, perform very similarly to each other and not as well. Looking at the FS results, the selected features of k-NN SFS and SVM SFS are very similar. They have the top five features in common (Balaban-J index, polarity number, Padmakar-Ivan index, Estrada index, and Narumi-Katayama index). This seems to indicate that some features are generally more important than others. Our ranking system seems to show that certain features, such as the mentioned top five features, have an impact on achieving higher accuracies. It seems that choosing the right features is important for more accurate classifications. In fact, these features may be important enough to always select them regardless of which classifier is used. This would be interesting to test by using the same five features for the ANN and see whether the results align with this assumption. These top five features are all topological descriptors and this may be an indication that important structural properties of graphs are preserved in the embedding process by the topological features. The extent of this preservation seems to be sufficient to achieve good classification accuracies. A possible flaw in our ranking system may be that there are four PTC data sets. There are only two MD and two mutagenic data sets. Performing aggregations over these imbalances may skew the results. However, when analyzing the features selected in the cases of SFS k-NN and SFS SVM, the top five features mentioned before are also heavily represented in the MD and mutagenic data sets. Due to this, we believe the results may only be skewed slightly, if at all. A further critique could be that we only employ our own ranking system and other ranking systems have not been explored. We acknowledge this, however, the FS study is not our primary focus in this thesis and therefore we do not investigate further options. Nevertheless, we believe it would be interesting to evaluate the features with other ranking systems. Additionally, focusing more on the data sets themselves, instead of aggregating them, could also be investigated further.

The times we present give some insight into the relationship between the computational complexity of the two approaches. However, testing shows that sometimes the computational time varies for exactly the same tasks (which is an unexpected phenomenon). This is due to having to use the servers provided by the university which we cannot fully control. Since this thesis mainly focuses on the accuracy comparison between the two approaches, and the time comparison is a subgoal, we do not investigate this further and present the times to give a rough understanding. Thus, while these times provide a valuable perspective and indicative overview, they should not be considered absolute values and must be interpreted with caution. A separate study focusing on this in more detail would need to be conducted to form precise conclusions. Additionally, a scalability study could be done. Despite this uncertainty, it seems that the TDE is a significantly faster approach compared to the GDE. It seems that the bigger the data set the more the computational time differs, as the difference is by far the biggest on the Mutagenicity data set. Regarding the comparison of SFS on the ANN model to the GCN, the GCN seems to perform faster, therefore negating the TDE's main benefit if FS is applied.

The proposed method exhibits some further potential limitations. Some graphs transform into the same feature vectors despite being different (e.g. in the PTC_MR and DHFR_MD data sets, there are 22 or 74 non-unique graphs, respectively). This could imply that information may be lost due to the embedding. However, it could also be that useless information is discarded by performing the embedding. This aspect is not thoroughly explored, and it would be insightful to determine which of the two possibilities applies. Another limitation is encountered during the embedding of the Mutagenicity data set. Some descriptors cannot deal with not connected graphs (e.g. the Wiener index results in infinity). One approach to fixing this could be applying the descriptors to the different components of the not connected graphs and using their average. Despite these limitations, we believe that the TDE proves its potential and we believe our hypothesis (the TDE can compete with modern classification methods while being computationally faster) proves itself to be partially true. It shows its strength when going up against the GED, achieving comparable accuracy while offering a substantial increase in speed.

Several of our results are comparable to those of Li et al. [27]. The five data sets that both this thesis and their paper have in common are the PTC data sets and the MUTAG data set. We need to be wary of some key differences in the experiment methods. The embedding they perform uses 20 descriptors and only four of them overlap with our 17. This difference in descriptors may have an impact on the accuracies. They only use an SVM instead of three different classifiers, and rather than reporting one accuracy for their performance assessment, they report the average accuracy and standard deviation over a tenfold CV run for each data set. With these differences in mind, we now compare our accuracies to theirs. Just as in our study, they achieve a statistically significant improvement on the MUTAG data set, further strengthening our belief that this method can be a competitive and perhaps even a stronger alternative to GED in some cases. On the PTC data sets, we are better four out of four times with the TDE method, whereas they are only better one out of four times (with z-normalization). The WL kernel beats their embedding the other three times. Therefore, we believe it may be interesting to compare our TDE to a WL kernel and see how the TDE compares to it. The GED may have a weakness in these types of data sets and it would be interesting to compare the TDE to a stronger reference system. When comparing their feature study to ours, different features seem to be picked more frequently than in our feature study. When analyzing the top five features selected by our SVM with SFS, we find very little variance. Only seven different features make up the top five over all five data sets. In comparison, 13 different features appear in the top five in their study. We believe they might experience higher variance in the top five features because they use more features, and their features might be more uniformly significant compared to ours.

Chapter 6

Conclusions and Future Work

This chapter contains a summary of the method and the most relevant results. We form our conclusions and present some limitations of the thesis, before touching on possible future work. The author presents his personal opinion in the last paragraph.

The proposed embedding method belongs to the graph-based PR methodologies. It uses 14 different topological and three different label descriptors to perform an embedding of graph data sets, turning them into data sets containing vectors in the feature space $\mathbb{R}^{17}$. Three models are used (k-NN, SVM, and ANN) for classification. The reference systems for the k-NN and SVM models employ the GED, while a GCN is used as a reference for the ANN. Furthermore, the application of two FS methods (SFS and mRMR) in combination with TDE is studied. After z-normalization and a potential FS, TDE achieves statistically superior or comparable accuracies when the models SVM and k-NN are used. It accomplishes this with significantly reduced computational time. The ANN model combined with our approach does not perform as well. Not only are there statistically inferior accuracies when compared to the GCN, but the main benefit of using the TDE (less computational complexity) is less prominent as with the other two models or is even eliminated (when SFS is combined with the ANN). Therefore, we conclude that the proposed embedding seems to be an alternative approach to the reference systems applying GED due to its lower computational complexity and competitive or improved accuracies. Conversely, the GCNs prove themselves stronger than the TDE combined with an ANN. Consequently, we conclude that our hypothesis is partially verified. While our approach can compete with a modern classification method (GED) when certain models are applied (k-NN or SVM), the approach does not seem to be able to compete across all types of methods (e.g. neural networks).

The feature study we conduct indicates that features play essential roles in classification. We believe that certain descriptors may capture vital structural properties of graphs (e.g. the Balaban-J index) and if these descriptors were to be further investigated or even modified and improved, this could consequently improve the performance of TDE. Although we believe it is likely that descriptors exhibit varying influences rather than being of universally equal importance (they may depend on the classifier or data set), our study seems to indicate that certain descriptors may be universal to some extent. Certain descriptors might be universally effective for specific types of graph data, such as molecular data sets, given that our feature study indicates that certain descriptors benefit all the examined data sets.

Despite the promise shown by our proposed method it is important to acknowledge its limitations. While our method demonstrates competitive performance across various data sets, it is not yet clear how it would perform on data sets that are fundamentally different or more complex than molecular ones. The adaptability to such scenarios and the scalability of our method remain unanswered questions. Furthermore, the effectiveness of our method seems to depend on which descriptors are used, which may be a difficult factor to optimize. Additionally, this thesis only compares fairly simple classification models that could be optimized more. One could perform experiments using state-of-the-art models for both the TDE and the GED reference to assess if TDE still manages to achieve comparable or better results in such a setting. Since both the GED and TDE models were always equally optimized we do not expect the results to change fundamentally.Additional future work could include exploring how versatile this method is. We believe that this approach may apply to other types of graph data sets since it proves itself to be competitive across eight different data sets in this thesis. The FS methods indicate that certain features are more important than others. Therefore, one may simply be able to modify or remove the badly performing descriptors entirely, as long as enough other descriptors are still available. We believe this to be an opportunity for this approach. However, as of now this is just an assumption and would need further investigation. Exploring this flexibility could also be done by applying it to new types of graph data sets, not only molecular ones. Furthermore, it would also be interesting to compare this method to other methods (e.g. Weisfeiler-Lehman kernel) with the SVM and k-NN classifiers.

In this final paragraph I would like to briefly present my thoughts and assumptions. I think that the proposed method is a valid alternative to other approaches which tackle the classification problem in PR, especially when computational efficiency is of the essence. I believe it makes sense to pair it with a simple classifier like k-NN. Since a substantial advantage of the TDE is its capability to handle large amounts of data efficiently, it could be advantageous to make use of this method as a baseline method. It could compute minimal first accuracies on data to provide insights, before employing other more complex models. This way it could be used to filter or categorize data as a first step. I believe this way TDE may find applications in the real world. Furthermore, it seems to me that using the optimal descriptors is key to achieving high accuracies and improving them could lead to even better results. This belief is influenced by an unexpected finding: an initial error in the Balaban-J index implementation which, interestingly, boosted the accuracy across all classifiers and data sets. It was only a minor modification that seemed to have a significant impact. That is why I believe further investigation of the descriptors and understanding them in more detail could improve this method even further, making it even more viable.

Appendix A

Algorithms

A.1 mRMR Algorithm

Algorithm 1 mRMR Algorithm

```
   Input: $X$: Data, $y$: Labels, $k$: number of features
   Output: selected_features[ ]
 1: selected_features = [ ]
 2: not_selected_features = X.features.to_list()
 3: for $i = 1, ..., k$ do
 4:    f_scores = [ ]
 5:    for feature $f \in$ not_selected_features do
 6:
 7:       f_score = $\frac{relevance(X[f],y)}{redundancy(X[f],X[selected\_features])}$
 8:
 9:       f_scores.append(f_score)
10:    end for
11:    best_feature = not_selected_features[f_scores.argmax()]
12:    selected_features.append(best_feature)
13:    not_selected_features.remove(best_feature)
14: end for
15: return selected_features
```

A.2 SFS Algorithm

Algorithm 2 SFS Algorithm

```
    Input: $X$: Data, $y$: Labels, clf: classifier $k$: number of features
    Output: selected_features[ ]
 1: selected_features = [ ]
 2: not_selected_features = X.features.to_list()
 3: for $i = 1, ..., k$ do
 4:     f_scores = [ ]
 5:     for feature $f \in$ not_selected_features do
 6:         f_score = cv_score(clf, $X$, $y$)
 7:         f_scores.append(f_score)
 8:     end for
 9:     best_feature = not_selected_features[f_scores.argmax()]
10:     selected_features.append(best_feature)
11:     not_selected_features.remove(best_feature)
12: end for
13: return selected_features
```

Appendix B

Hyperparameters

The radial basis function is abbreviated with RBF in the following tables.

B.1 Reference Hyperparameters Results

Classifier	**Metric**	**MUTAG**	**Mutagenicity**	**ER_MD**	**DHFR_MD**	**PTC_MR**	**PTC_MM**	**PTC_FM**	**PTC_FR**
k-NN	α	0.35	0.05	0.45	0.35	0.35	0.45	0.05	0.25
	k	18	1	1	2	5	15	23	4
	algorithm	brute	brute	brute	brute	brute	brute	brute	brute
SVM	α	0.45	0.45	0.45	0.45	0.45	0.15	0.25	0.35
	C	0.1	0.01	0.01	0.001	0.1	0.001	0.001	0.001

Table B.1: Reference Hyperparameters Results

B.2 TDE Hyperparameters Results

Classifier	**Metric**	**MUTAG**	**Mutagenicity**	**ER_MD**	**DHFR_MD**	**PTC_MR**	**PTC_MM**	**PTC_FM**	**PTC_FR**
k-NN	k	1	9	1	5	3	8	8	4
	metric	euclidean	cosine	manhattan	euclidean	manhattan	euclidean	manhattan	cosine
	algorithm	brute	brute	brute	brute	brute	brute	brute	brute
SVM	C	100	1	10	10	100	100	1	10
	γ	0.1	1	100	1	10	1	100	10
	kernel	RBF	RBF	RBF	RBF	RBF	RBF	RBF	RBF

Table B.2: TDE Hyperparameters Results with FS

Classifier	**Metric**	**MUTAG**	**Mutagenicity**	**ER_MD**	**DHFR_MD**	**PTC_MR**	**PTC_MM**	**PTC_FM**	**PTC_FR**
	k	1	12	2	3	1	6	8	16
k-NN	metric	euclidean	cosine	cosine	cosine	manhattan	cosine	euclidean	manhattan
	algorithm	brute	brute	brute	brute	brute	brute	brute	brute
	C	100	100	100	10	10	1	1	1
SVM	γ	0.01	0.1	10	100	10	100	100	100
	kernel	RBF	RBF	RBF	RBF	RBF	RBF	RBF	RBF

Table B.3: TDE Hyperparameters Results without FS

Bibliography

[1] Ashok K. Goel and Jim Davies. *Artificial Intelligence*, page 468–482. Cambridge Handbooks in Psychology. Cambridge University Press, 2011.

[2] Allen Newell and Herbert Simon. The logic theory machine–a complex information processing system. *IRE Transactions on information theory*, 2(3):61–79, 1956.

[3] Emma S. Brunette, Rory C. Flemmer, and Claire L. Flemmer. A review of artificial intelligence. In *2009 4th International Conference on Autonomous Robots and Agents*, pages 385–392, 2009.

[4] Ioannis Konstas, Vassilios Stathopoulos, and Joemon M Jose. On social networks and collaborative recommendation. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, pages 195–202, 2009.

[5] Siyue Xie, Haifeng Hu, and Yongbo Wu. Deep multi-path convolutional neural network joint with salient region attention for facial expression recognition. *Pattern recognition*, 92:177–191, 2019.

[6] Peter Flach. *Machine Learning: The Art and Science of Algorithms that Make Sense of Data*. Cambridge University Press, 2012.

[7] Christopher M Bishop and Nasser M Nasrabadi. *Pattern recognition and machine learning*, volume 4. Springer, 2006.

[8] Horst Bunke and Kaspar Riesen. Recent advances in graph-based pattern recognition with applications in document analysis. *Pattern Recognition*, 44(5):1057–1067, 2011.

[9] HongYun Cai, Vincent W. Zheng, and Kevin Chen-Chuan Chang. A comprehensive survey of graph embedding: Problems, techniques, and applications. *IEEE Transactions on Knowledge and Data Engineering*, 30(9):1616–1637, 2018.

[10] Girish Chandrashekar and Ferat Sahin. A survey on feature selection methods. *Computers Electrical Engineering*, 40(1):16–28, 2014.

[11] Kaspar Riesen and Horst Bunke. *Graph Classification and Clustering Based on Vector Space Embedding*, volume 77 of *Series in Machine Perception and Artificial Intelligence*. WorldScientific, 2010.

[12] Vito Latora, Vincenzo Nicosia, and Giovanni Russo. *Graphs and Graph Theory*, page 1–30. Cambridge University Press, 2017.

[13] Sakander Hayat, Shaohui Wang, and Jia-Bao Liu. Valency-based topological descriptors of chemical networks and their applications. *Applied Mathematical Modelling*, 60:164–178, 2018.

[14] Ivan Gutman, B Ruščić, Nenad Trinajstić, and Charles F Wilcox Jr. Graph theory and molecular orbitals. xii. acyclic polyenes. *The journal of chemical physics*, 62(9):3399–3405, 1975.

[15] Hideyuki Narumi and Meiseki Katayama. Simple topological index: a newly devised index characterizing the topological nature of structural isomers of saturated hydrocarbons. *Memoirs of the Faculty of Engineering, Hokkaido University*, 16(3):209–214, 1984.

[16] Harry Wiener. Relation of the physical properties of the isomeric alkanes to molecular structure. surface tension, specific dispersion, and critical solution temperature in aniline. *The Journal of Physical Chemistry*, 52(6):1082–1089, 1948.

[17] Mehdi Eliasi, Ghaffar Raeisi, and Bijan Taeri. Wiener index of some graph operations. *Discrete Applied Mathematics*, 160(9):1333–1344, 2012.

[18] Milan Randic. Characterization of molecular branching. *Journal of the American Chemical Society*, 97(23):6609–6615, 1975.

[19] Ernesto Estrada. Characterization of 3d molecular structure. *Chemical Physics Letters*, 319(5-6):713–718, 2000.

[20] Ivan Gutman, Slavko Radenković, Ante Graovac, and Dejan Plavšić. Monte carlo approach to estrada index. *Chemical Physics Letters*, 446(1-3):233–236, 2007.

[21] Alexandru T Balaban. Highly discriminating distance based numerical descriptor. *Chem. Phys. Lett*, 89:399–404, 1982.

[22] Padmakar V Khadikar, Narayan V Deshpande, Prabhakar P Kale, Andrey Dobrynin, Ivan Gutman, and Gyula Domotor. The szeged index and an analogy with the wiener index. *Journal of Chemical Information and Computer Sciences*, 35(3):547–550, 1995.

[23] Padmakar V Khadikar. On a novel structural descriptor pi. *National Academy Science Letters*, 23:113–118, 2000.

[24] Harry P Schultz. Topological organic chemistry. 1. graph theory and topological indices of alkanes. *Journal of Chemical Information and Computer Sciences*, 29(3):227–228, 1989.

[25] Shazia Manzoor, Muhammad Kamran Siddiqui, and Sarfraz Ahmad. On entropy measures of molecular graphs using topological indices. *Arabian Journal of Chemistry*, 13(8):6285–6298, 2020.

[26] MH Khalifeh, MR Darafsheh, and Hassan Jolany. The hyper wiener index of one pentagonal carbon nanocone. *arXiv preprint arXiv:1212.4411*, 2012.

[27] Geng Li, Murat Semerci, Bülent Yener, and Mohammed J. Zaki. Effective graph classification based on topological and label attributes. *Statistical Analysis and Data Mining: The ASA Data Science Journal*, 5(4):265–283, 2012.

[28] Thomas Cover and Peter Hart. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13(1):21–27, 1967.

[29] Vladimir Vapnik. *The nature of statistical learning theory*. Springer science & business media, 1999.

[30] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986.

[31] N Hema Rajini and R Bhavani. Automatic classification of computed tomography brain images using ann, k-nn and svm. *AI & society*, 29:97–102, 2014.

[32] László Lovász. Random walks on graphs. *Combinatorics, Paul erdos is eighty*, 2(1-46):4, 1993.

[33] 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)*, 2020.

[34] Nils Kriege and Petra Mutzel. Subgraph matching kernels for attributed graphs. *arXiv preprint arXiv:1206.6483*, 2012.

[35] Asim Kumar Debnath, Rosa L. Lopez de Compadre, Gargi Debnath, Alan J. Shusterman, and Corwin Hansch. Structure-activity relationship of mutagenic aromatic and heteroaromatic nitro compounds. correlation with molecular orbital energies and hydrophobicity. *Journal of Medicinal Chemistry*, 34(2):786–797, 1991.

[36] Jeroen Kazius, Ross McGuire, and Roberta Bursi. Derivation and validation of toxicophores for mutagenicity prediction. *Journal of Medicinal Chemistry*, 48(1):312–320, 2005.

[37] Kaspar Riesen and Horst Bunke. Iam graph database repository for graph based pattern recognition and machine learning. In Niels da Vitoria Lobo, Takis Kasparis, Fabio Roli, James T. Kwok, Michael Georgiopoulos, Georgios C. Anagnostopoulos, and Marco Loog, editors, *Structural, Syntactic, and Statistical Pattern Recognition*, pages 287–297, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg.

[38] Jeffrey J. Sutherland, Lee A. O'Brien, and Donald F. Weaver. Spline-fitting with a genetic algorithm: a method for developing classification structureactivity relationships. *Journal of Chemical Information and Computer Sciences*, 43(6):1906–1915, 2003.

[39] Chris Ding and Hanchuan Peng. Minimum redundancy feature selection from microarray gene expression data. In *Computational Systems Bioinformatics. CSB2003. Proceedings of the 2003 IEEE Bioinformatics Conference. CSB2003*, pages 523–528, 2003.

[40] A. Wayne Whitney. A direct method of nonparametric measurement selection. *IEEE Transactions on Computers*, C-20(9):1100–1103, 1971.

[41] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Andreas Müller, Joel Nothman, Gilles Louppe, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[42] Michel Neuhaus and Horst Bunke. *Bridging the Gap Between Graph Edit Distance and Kernel Machines.* World Scientific Publishing Co., Inc., USA, 2007.

[43] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[44] Guido Van Rossum and Fred L. Drake. *Python 3 Reference Manual.* CreateSpace, Scotts Valley, CA, 2009.

[45] Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with NumPy. *Nature*, 585(7825):357–362, September 2020.

[46] The pandas development team. pandas-dev/pandas: Pandas, February 2020.

[47] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

Erklärung

gemäss Art. 30 RSL Phil.-nat.18

Name/Vorname:	Gibson Robin
Matrikelnummer:	20-119-467
Studiengang:	Informatik
	Bachelor [✓] Master ☐ Dissertation ☐
Titel der Arbeit:	Graph Representation Through Topological Descriptors
LeiterIn der Arbeit:	PD Dr. Kaspar Riesen

Ich erkläre hiermit, dass ich diese Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen benutzt habe. Alle Stellen, die wörtlich oder sinngemäss aus Quellen entnommen wurden, habe ich als solche gekennzeichnet. Mir ist bekannt, dass andernfalls der Senat gemäss Artikel 36 Absatz 1 Buchstabe r des Gesetzes vom 5. September 1996 über die Universität zum Entzug des auf Grund dieser Arbeit verliehenen Titels berechtigt ist.
Für die Zwecke der Begutachtung und der Überprüfung der Einhaltung der Selbständigkeitserklärung bzw. der Reglemente betreffend Plagiate erteile ich der Universität Bern das Recht, die dazu erforderlichen Personendaten zu bearbeiten und Nutzungshandlungen vorzunehmen, insbesondere die schriftliche Arbeit zu vervielfältigen und dauerhaft in einer Datenbank zu speichern sowie diese zur Überprüfung von Arbeiten Dritter zu verwenden oder hierzu zur Verfügung zu stellen.

Dübendorf, 17.08.2023

Ort/Datum

[illegible]

Unterschrift