dortmund university

Providing Bipartite GNN Explanations with PGExplainer

Tristan Marten Lewin Schulz

Bachelor of Science

May 19, 2025

Supervisors:
Prof. Dr. Stefan Harmeling
Lukas Schneider

Artificial Intelligence (VIII)
Department of Computer Science
TU Dortmund University

Abstract

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua.

Contents

Chapter 1

Introduction

1.1 Motivation and Background

MOTIVATION SAT! Advancements in deep learning SAT solving, e.g. NeuroSAT. Need for evaluation of these models, use of GNN explainers as SAT can be reduced to graph domain. Application of PGExplainer on NeruoSAT to generate explanations. Explanations need gt to be evaluated on accuracy. Use concepts like unsat cores and backbones as gt and compare to explanations provided by PGExplainer to see whether NeuroSAT explanations align with "human-observable" principles. Therefore explanation and replication of PGExplainer. After successful replication, application on NeuroSAT. Graph

task: Prediction of UNSAT and unsat cores as gt. Node task: Difficult with NeuroSAT?

1.2 Thesis structure?

1.3 Related Work

TODO: MOVE TO BEFORE CONCLUSION?! MOVE TO THEORETICAL BACKGROUND?!
Original PGExplainer? We seek to provide a reimplementation using PyG framework and reproduce the results. TODO: Move criticism to discussion? Existing reimplementation

of PGExplainer using PyG by Holdijk et al. Unable to achieve results of original implementation. Improvement on some datasets, considerable deterioration on BA-2Motif. Criticize approach of original and lacklustre documentation as well as differences between codebase and paper. We try own implementation to further improve results. We follow the configs used in the reimplementation but conduct sweeps ourselves around these and the original configs. Compare results to both implementations.
Taxonomy paper evaluates PGExplainer on Fidelty and achieves low scores, sides with reproducibility paper above: "is not performing as promising as its original reported results".

Propose only using PGExplainer for Node classification task.NeuroSAT as downstream task for SAT experiments with PGExplainer. PyG reimplementation code provided by Rodhi. We create required data with provided methods, add unsat cores as gt and adapt the NeuroSAT model to allow passing edge weights into the adjacency matrix. PGExplainer adapted for SAT data, NeuroSAT embeddings and evaluation.

Chapter 2

Background

In this chapter we define the necessary background for understanding PGExplainer as well as the follow-up work regarding its application on SAT.

2.1 Deep learning

What is machine learning? What is Deep learning?
Principle behind neural networks: Layers, gradients, gradient descent?, backprop...

2.1.1 MLP

2.1.2 Batchnorm, LayerNorm, ...

2.2 Graph Theory

These definitions will loosely follow Brachman et al.[**<empty citation>**]. A graph is a data structure consisting of a set of nodes that are connected via edges, modeling objects and their relationships. It can be represented as $G = (V, E)$ with $V = \{v_1, v_2...v_n\}$ being the set of n nodes, and $E \in V \times V$ the set of edges. An edge $e = (u, v)$ connects nodes u and v, making them neighbours. Edges are either directed or undirected and lead to directed or undirected graphs if exclusively present. The degree of a node v is the number of edges connected to v and denoted by $d(v)$. G can be described by an adjacency matrix $A \in \mathbb{R}^{n \times n}$, where

$$A_{ij} = \begin{cases} 1 & \text{if } \{v_i, v_j\} \in E \text{ and } i \neq j, \\ 0 & \text{otherwise.} \end{cases}$$

If G is an undirected Graph the adjacency matrix will be symmetrical. We adopt the conventions from Diestel[4] to refer to the node and edge set of any graph G with $V(G)$ and $E(G)$, regardless of the actual names of the sets, as well as a referring to G with node set V as G on V. G is called a subgraph of another graph $G' = (V', E')$ if $V(G) \subseteq V(G')$ and $E(G) \subseteq E(G')$. This is denoted as $H \subseteq G$. The number of nodes in a graph $|V|$

is its order and the number of edges $|E|$ is its size. We additionally define bipartite graphs according to Asratian et al.[1]: A graph G is bipartite if the set of nodes V can be partitioned into two sets V_1 and V_2 so that no two nodes from the same set are adjacent. The sets V_1 and V_2 are called colour classes and (V_1, V_2) is a bipartition of G. This means that if a graph is bipartite all nodes in V can be coloured by at most two colours so that no two adjacent nodes share the same colour.
TODO: EDGE WEIGHTS, K-hop/computational graph?

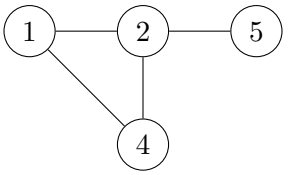

Figure 2.1: A simple undirected graph G with $V = \{1, ..., 5\}$ and $E = \{\{1, 2\}, \{2, 4\}, \{1, 4\}, \{2, 5\}\}$

2.2.1 Graph generation/Graph Generative Model/Random Graphs

Erdos-Renyi model first model of random graphs, but slightly different from Gilbert.

TODO: WRONG! (Independent experiments with same probability!) Gilbert[**<empty citation>**] describes the process of generating a random graph G of order N by assigning independent probabilities p to each potential edge between two nodes in $V(G)$ to exist in $E(G)$(, effectively drawing from a Bernoulli distribution).
A random graph is further described by Diestel[**<empty citation>**] as follows. Let $V = \{0, ..., n-1\}$ be a fixed set of n elements. Our goal is to define the set $\mathcal{G}$ of all graphs on V as a probability space, which allows us to ask whether a Graph $G \in \mathcal{G}$ has a certain property. To generate our random graph we then decide from some random experiment whether e shall be an edge of G for each potential $e \in [V]^2$. The probability of success - accepting e as edge in G - is defined as $p \in [0, 1]$ for each experiment. TODO: Rewrite the accepting part. This leads to the probability of G being a particualar graph G_0 on V with e.g. m edges being equal to $p^m q^{\binom{n}{2}-m}$ with $q := 1 - p$. It follows our desired probability space $\mathcal{G} = (n, p)$ as the product space

$$\Omega := \prod_{e \in [V]^2} \Omega_e$$

with $\Omega_e := \{0_e, 1_e\}$, $\mathbb{P}_e(\{1_e\}) := p$ and $\mathbb{P}_e(\{0_e\}) := q$.

$$E(G) = \{e | \omega(e) = 1_e\}$$

G is called a random graph on V with egde probability p.

TODO: Define $[V]^2$ as the set of all 2-elements subsets of V? or rewrite as e in E x E
INSTEAD: Gilberts idea also assigns the same probability across all edges, so probably best to explain the general idea as presented in Diestel. Then explain the difference in PGE? PGEs appraoch mainly inspired by probabilistic graphical model/bayesian networks. Gilbert model mainly baseline for probabilistic graphs. PGExplainer mainly inspired by Gilbert, but "required" concept is PGM.

2.3 Information Theory

To fully understand the learning objective of PGExplainer it is necessary to define the concepts of entropy and mutual information. We follow the definitions by Cover et al.[3][p.13].

2.3.1 Entropy

Entropy is used to describe the uncertainty of a random variable. Let X be a discrete random variable with alphabet $\mathcal{X}$ and probability mass function $p(x) = Pr\{X = x\}$ for $x \in X$. The entropy $H(X)$, also written as $H(p)$ is defined as

$$H(X) = -\sum_{x \in \mathcal{X}} p(x) \log p(x). \tag{2.1}$$

The log is to the base 2 and entropy is measured in bits. TODO: DEFINE EDGE CASES log 0 TODO: TOO MUTCH + SOURCE? A simple example is tossing two coins: There are four possible outcomes $\mathcal{X} = \{00, 10, 01, 11\}$, each with a probability $p = 0,25$. The resulting entropy $H(X) = 2$ represents that two bits of information can be stored this way.

TODO: NOT USED IN PGE, ONLY FOR DEFINITION OF COND. ENT. IF KEPT, DIFFER MORE CLEARLY FROM CROSS?
Analogously we define the joint entropy $H(X, Y)$ of a pair of discrete random variables (X, Y) with a joint distribution $p(x, y)$ as follows:

$$H(X, Y) = -\sum_{x \in \mathcal{X}} \sum_{y \in \mathcal{Y}} p(x, y) \log p(x, y). \tag{2.2}$$

TODO: REWRITE! MAYBE NOT NEEDED AS NOT "USED" IN PGE?
The conditional entropy of Y given X is defined as the expected value of the entropies of the conditional distributions, averaged over the conditioning random variable. If $(X, Y) \sim p(x, y)$ the conditional entropy is defined as

$$\begin{aligned} H(Y|X) &= -\sum_{x \in \mathcal{X}} p(x) H(Y|X = x) && (2.3)\\ &= -\sum_{x \in \mathcal{X}} \sum_{y \in \mathcal{Y}} p(x, y) \log p(y|x) && (2.4)\\ &= -E \log p(Y|X) \text{ with E = Expectation.} && (2.5) \end{aligned}$$

Elements of Information Theory: equation 2.26 describes KL distance/relative entropy

TODO: KL distance or relative entropy with probability mass functions $p(x), q(x)$:

$$D_{KL}(p||q) = \sum_{x \in \mathcal{X}} p(x) \log \frac{p(x)}{q(x)} \tag{2.6}$$

Cross entropy by Goodfellow et al.[5][p.75] assumes probability distributions P, Q:

$$H(P, Q) = -\mathbb{E}_{x \sim P} \log Q(x) \tag{2.7}$$

Closely related to KL divergence and can therefore also be expressed as:

$$H(P, Q) = H(P) + D_{KL}(P||Q) \tag{2.8}$$

We derive for the discrete case with mass probability functions p, q with the same support $\mathcal{X}$ (TODO: NOT NEEDED? for random Variables X, Y):

$$H(p, q) = H(p) + D_{KL}(p||q) = -\sum_{x \in \mathcal{X}} p(x) \log p(x) + \sum_{x \in \mathcal{X}} p(x) \log \frac{p(x)}{q(x)} \tag{2.9}$$

$$= -\sum_{x \in \mathcal{X}} p(x) \log p(x) + \sum_{x \in \mathcal{X}} p(x) \log p(x) - \sum_{x \in \mathcal{X}} p(x) \log q(x) \tag{2.10}$$

$$= -\sum_{x \in \mathcal{X}} p(x) \log q(x) \tag{2.11}$$

The approach in PGExplainer is a common approach in ML for simplifying objectives? FIND LITERATURE THAT EXPLAINS APPROXIMATION OF COND. ENTROPY WITH CROSS ENTROPY

"We can modify the conditional entropy objective in Equation 4 with a cross entropy objective between the label class and the model prediction" (GNNExplainer)

2.3.2 Mutual Information

Another closely related concept is mutual information (see Cover et al.[3][p.19]). It measures the amount of information that one random variable contains about another or the reduction in uncertainty of said variable due to knowing the other. Let X and Y be two random variables with the joint probability mass function $p(x, y)$ and marginal probability mass functions $p(x)$ and $p(y)$. Mutual information $I(X; Y)$ is the relative entropy between the joint distribution and product distribution $p(x)p(y)$:

$$I(X; Y) = \sum_{x \in \mathcal{X}} \sum_{y \in \mathcal{Y}} p(x, y) \log \frac{p(x, y)}{p(x)p(y)} \tag{2.12}$$

$$= H(X) - H(X|Y) \tag{2.13}$$

2.3.3 Monte Carlo Sampling

Goodfellow et al.[5][p.590] TODO: EXPLANATION
Let

$$s = \sum_x p(x)f(x) = E_p[f(x)] \tag{2.14}$$

be the sum to estimate with p being a probability distribution over a random variable x. Then s can be approximated by drawing n samples from p and constructing the empirical average

$$\hat{s}_n = \frac{1}{n}\sum_{i=1}^{n} f(x^{(i)}). \tag{2.15}$$

2.4 Graph Neural Networks

The following definitions will loosely follow book/... (reference). JEDE Variable einmal erklärt! Einheitliche Variablen aus meiner Sicht
Graph Neural Networks(GNNs)[9] are a deep learning-based approach that operates on graphs, a data structure consisting of nodes and edges, representing objects and their relationships. Due to their unique non-Euclidean property, they find usage in classification, link prediction, and clustering tasks. Their high interpretability and strong performance have led to GNNs becoming a commonly employed method in graph analysis. They combine the key features of convolutional neural networks[7], such as local connection, shared weights, and multi-layer usage, with the concept of graph embeddings[2] to leverage the power of feature extraction and representation as low-dimensional vectors for graphs[8].
- difference node task, graph tasks (Scarselli)
- data preprocessed: mapping to simpler representation - encode graph structure/topology to keep structural information - directed/undirected? - ...

2.4.1 Convolutional Graph Neural Networks

Explain? Used in architecture of downstream task, only slightly relevant

2.5 Perturbation-based Explainability in GNNs

Explainability methods in the context of deep learning and further GNNs
Different approaches, further into perturbation based...

- Common metric to measure accuracy of explanation methods in comparison to ground truth is ROC-AUC score? (Explain/name first in results?) - Other metric includes fidelity, results of taxonomy propose only using PGExplainer for Node Classification as it achieves low fidelity on Graph tasks

2.6 Boolean Satisfiability Problem

We define the Boolean Satisfiability Problem (SAT) according to Guo et al.[6][p.641]:
A Boolean formula is constructed from Boolean variables, that only evaluate to True (1) or False (0), and the three logic operators conjunction ($\wedge$), disjunction ($\vee$) and negation ($\neg$). SAT aims to evaluate whether there exists a variable assignment for a formula constructed of said parts so that it evaluates to True. If so, the formula is said to be satisfiable or unsatisfiable otherwise. Every propositional formula can be converted into an equivalent formula in conjunctive normal form (CNF), which consists of a conjunction of one or more clauses. These clauses must contain only disjunctions of at least one literal (a variable or its negation). In this work we consider only formulas in CNF, as NeuroSAT[**<empty citation>**] assumes SAT problems to be in CNF. An example of a satisfiable formula in CNF over the set of variables $V = \{x_1, x_2\}$ is

$$\psi(V) = (x_1) \wedge (\neg x_1 \vee x_2) \wedge (\neg x_2 \vee x_2)$$

with satisfying assignment $A : \{x_1 \mapsto 1, x_2 \mapsto 1\}$. Furthermore, SAT is NP-complete, meaning that if there exists a deterministic algorithm able to solve SAT in polynomial time, then such an algorithm exists for every NP problem.

2.6.1 Representation as Bipartite Graph

From Guo: 4 types of graph representations for CNF formulae: LCG, LIG, VCG, VIG. "LCG is a bipartite graph with literals on one side and clauses on the other side, with edges connecting literals to the clauses where they occur." Representation of SAT in context of NeuroSAT: literal-clause graph (LCG). Messages are passed between clauses and literals, as well as literals and their complement. 1. Clause receives from neighboring literals 2. Literals receive from clauses and complement.
Formal definition as incident graph?
TODO: TOO BIG

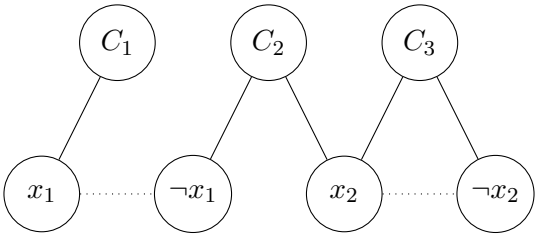

Figure 2.2: LCG representation for $\psi(V)$

2.6.2 Incidence/Levi graph?

Defined in ALYAHYA et al. Concrete graphical representation of SAT? Type of bipartite graph. Defines edges via edge weight function! PART OF GRAPH THEORY

Definition in Cimatti et al. : For clause c we use $lit(c)$ and $var(c)$ to reference the set of literals and variables in c respectively. "$For a CNF formula F we write cla(F) for its set of clauses, lit(F) = c \in cl$
Incidence graph of ψ is the bipartite graph $inc(\psi) = (V, E)$ with $V = lit(\psi) \cup cla(\psi)$. Additionally for literal $x \in lit(\psi)$ and clause $c \in cla(\psi)$ we define $xc \in E$ if $x \in var(c)$.
ALTERNATIVE: Describe levi graph shortly. Then define convention for our definitions of sat and the respective incidence graph that follows?

2.6.3 UNSAT Cores

Instances of SAT are commonly analyzed as graphs

2.6.4 Backdoors

Definition in Cimatti et al.

2.7 NeuroSAT

Chapter 3

PGExplainer - Main part

V1: In the following chapter, we introduce the PGExplainer[**<empty citation>**] and its concepts. The idea is to generate explanations in the form of probabilistic graph generative models for any learned GNN model, henceforth referred to as the downstream task (DT), by utilizing a deep neural network to parameterize the generation process. This approach seeks to explain multiple instances collectively, as they share the same neural network parameters, and therefore improve the generalizability to previous works, particularly the GNNExplainer[**<empty citation>**].

V2: In the following chapter, we introduce the PGExplainer[**<empty citation>**] and its concepts. The idea is to generate explanations in the form of probabilistic graph generative models that have proven to learn the concise underlying structures of GNNs most relevant to their predictions. This approach may be applied to any learned GNN model, henceforth referred to as the downstream task (DT), by utilizing a deep neural network to parameterize the generation process. PGExplainer seeks to explain multiple instances collectively, as they share the same neural network parameters, and therefore improve the generalizability to previous works, particularly the GNNExplainer[**<empty citation>**]. This means that all edges in the dataset are predicted by the same model, which leads to a global understanding of the DT.

Transductive/Inductive relevant?

TODO: Maybe separate the theory of PGE from our own work more strictly? E.g. 3. PGE theory, 4. PGE reimplementation and NeuroSAT application?

We then describe our reimplementation in detail (Section 3.2), including the changes made and difficulties during the process.

In Section 3.3 we present the idea of applying PGExplainer on the NeuroSAT framework to generate explanations for the machine learning SAT-solving approach and comparing these to "human-understandable" concepts like UNSAT cores and backbone variables.

3.1 Theory

We follow the structure of the original Paper[**<empty citation>**] and start by describing the learning objective (Section 3.1.1), the reparameterization trick (Section 3.1.2), the global explanations (Section 3.1.3) and regularization terms (Section 3.1.4).

3.1.1 Learning Objective

3.1.2 Reparameterization Trick

3.1.3 Global Explanations

3.1.4 Regularization Terms

3.2 Reimplementation

Implementation details:

- Started by reimplementing the downstream tasks used in og paper for node and graph class.
- Node datasets taken from original and transformed to a "pyg format", to keep original structure and ground truths
- Graph: BA-2Motif from pyg library with self generated gt, MUTAG dataset taken from pyg and added gt-labels that were added in original dataset
- Created pytorch/pytorch geometric implementation of 3-layer graph conv networks
- architecture as described in paper: ReLu activation, Pooling for graph net
- Xavier initialization for all layers
- Same hyperparameters?(Adam, $1 * 10^{-}3$ lr, 1000 epochs)/experimental setup?
- ADDED Dropout(0.1) to improve performance/overfitting on node tasks
- Fully connected layer: torch geometric GraphConv to allow passing of edge weights(+ bipartite)!?
- Original references sageConv in paper, pyG impl. does not allow edge weights, verify!
- GATConv for Graph attention(referenced by original)

- Alternatives: GCNConv(edge weights, no bipartite graphs)
- 80/10/10 split
- =¿ Similar accuracies achieved

Explainer: This is irrelevant for paper, description of why reparametrization trick necessary - First approach tried passing a masked edge index to downstream task with edge weights ¿ 0.5 =¿ Unable to learn with hard cut-off (bad for gradients). (Same with TopK probably?!) - Pass calculated edge weights to downstream task for learning. If no edge weights are passed(downstream task), all edge weights are initialized with one to represent all edges beeing relevant

3.3 Application on NeuroSAT

Reimplementation of NeuroSAT provided by Rodhi. As the code used for NeuroSAT can also be found in our Repository, we stress that only the changes described in the following chapter are part of our work.

What did we do? What did we change for NeuroSAT? What data was used? How did we adapt PGExplainer?

Only change in NeuroSAT: pass edge weights into adjacency matrix. Calculates ....
Generated batches of unsat problems that "turned" unsat because of last added clause. 10 literals per problem. Only unsat to test for unsat cores, that only apply for unsat problems. Calculated unsat cores with solver xy by adding negative assumption literals per clause and passing these as assumption for calulation. The edges of the clauses present in the unsat core were treated as ground truth.

Changes for explainer: Edge embeddings calced by DS and passed to explainer. Calculated edge embeddings by concatenating node embeddings for connected nodes, similar to original. Embeddings fed into MLP, weights sampled with reparam. trick to get edge "probabilities", passed as "unbatched" sampled graph into NeuroSAT predictor. Visualization of SAT problems with edge weights, ands gts.
For quant. eval. adapted roc auc as metric as done in PGExplainer. Results seem "good" but qual. eval. shows different result. roc auc bad metric?
For qual. eval. topk(=number of edges in gt) edges of predictions were highlighted to be compared to gt edges. For quant. eval. the edge probabilites were compared to gt with 1s for edges in gt and 0s for rest.

Chapter 4

Results

Experimental Setup: We follow the experimental setup from the PGExplainer as closely as possible. Since the textual description refers to the setup from GNNExplainer and is lacking in some aspects, we extract the missing information from the codebase. As the hyperparameters are unclear or not comprehensible for some tasks we also draw information from the configs of PYTORCH REIMPL.

We use normalization in our downstream models, though it is not described in the paper, since it is used in the code. We also experimented with the effects of the use of normalization since it seems to be relevant to the performance of the explainer.
The explainer is trained and evaluated on the same data. We also run experiments with added train/test splits TODO!
Not all data is fed into the explainer: BA-Shapes: BA-Community: Tree-Cycles: "First"/to base graph attached node of each motif in graph is used. Tree-Grid: All Motif nodes are used BA-2Motif: All graphs are used MUTAG: Only the graphs with an available ground truth are used. GT exist for mutagenic graphs that have either chemical groups NH2 or NO2. We later discuss if these selections make sense and run experiments with different data selections.
Quantitative: 10 explainer runs on one downstream model; Calculate ROC-AUC over ALL graphs/nodes in each run; Qualitative: Original uses a threshold; we instead take the topK nodes according to the dataset/motif as an explanation
We also discuss if treating the number of k edges as a hyperparameter dependant on the downstream task makes sense and propose having the network learn it, to improve generalizability and allow the explainer to work on data with varying size.

CPU vs GPU:
It is important to highlight that our code achieved better and way more stable results for BA-2Motif when trained on a gpu instead of cpu.
Ba-Shapes, Tree-Cycles and MUTAG results achieved were identical.

Ba-Community and Tree-Grid achieved very slightly better results on CPU.

Sweeps: (Params ordered by importance)
BA-Shapes: higher size reg -¿ 0.1; lower entropy reg -¿ 0.01; lr and tT very low impact but slightly higher -¿ 0.01 and 5. Note that Loss curve jumps on most runs! (logical-sweep-94 and restful-sweep-92 have clean loss) TRY HIGHER SIZE REG AND LOWER ENTROPY REG

BA-Community: lr 0.0001 too low, not working -¿ 0.003; lower entropy -¿ 0.1; higher size -¿ 0.1; TRY MORE SEEDS, LR, EPOCHS?

Tree-Cycles: high lr -¿ 0.01 ; lower entropy reg -¿ 0.1/0.01; higher size reg -¿ 0.1/0.01 ; lower tT -¿ 1. TRY WITH MORE SEEDS FOR ENT, SIZE, TEMP? Confirmed higher size reg -¿ 0.1; lower entropy reg -¿ 0.01; temp really low impact, tendency higher. TRY 30 EPOCHS???

Tree-Grid: high lr -¿ 0.01; high size reg -¿ 1; higher entropy reg? -¿ 10/1, high tT -¿ 5. TRY MORE SEEDS FOR ENTROPY REG -¿ not quite clear, tendency lower; MAYBE EVEN HIGHER LR -¿ No

BA-2Motif: RUN ON GPU

MUTAG: Low lr -¿ 0.0003; low entropy reg(high impact, but highest AUC runs vary) -¿ 0.1; low tT -¿ 1; less epochs -¿ 20; low size reg -¿ 0.005(/0.01); Loss is messy and AUC seems to decrease over time! lr 0.0001 worse, entropy reg 0.1/0.01 has zero effect -¿ 0.1

Effects of selected motif nodes for Node task: Compare Tree-Grid/Tree-Cycles performance when using all/one node per motif...

Chapter 5

Discussion

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua.

Chapter 6

Conclusion

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua.

Bibliography

[1] A.S. Asratian, T.M.J. Denley, and R. Häggkvist. *Bipartite Graphs and their Applications*. Cambridge Tracts in Mathematics. Cambridge University Press, 1998. ISBN: 9781316582688. URL: https://books.google.de/books?id=l8fLCgAAQBAJ.

[2] Hongyun Cai, Vincent W Zheng, and Kevin Chen-Chuan Chang. "A comprehensive survey of graph embedding: Problems, techniques, and applications". In: *IEEE transactions on knowledge and data engineering* 30.9 (2018), pp. 1616–1637.

[3] T.M. Cover and J.A. Thomas. "Entropy, Relative Entropy, and Mutual Information". In: *Elements of Information Theory*. John Wiley & Sons, Ltd, 2005, pp. 13–55. ISBN: 9780471748823. DOI: https://doi.org/10.1002/047174882X.ch2. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/047174882X.ch2. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/047174882X.ch2.

[4] Reinhard Diestel. "The Basics". In: *Graph Theory*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2017, pp. 1–34. ISBN: 978-3-662-53622-3. DOI: 10.1007/978-3-662-53622-3_1. URL: https://doi.org/10.1007/978-3-662-53622-3_1.

[5] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. http://www.deeplearningbook.org. MIT Press, 2016.

[6] Wenxuan Guo et al. "Machine learning methods in solving the boolean satisfiability problem". In: *Machine Intelligence Research* 20.5 (2023), pp. 640–655.

[7] Y. Lecun et al. "Gradient-based learning applied to document recognition". In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324. DOI: 10.1109/5.726791.

[8] Zhiyuan Liu and Jie Zhou. "Introduction". In: *Introduction to Graph Neural Networks*. Cham: Springer International Publishing, 2020, pp. 1–3. ISBN: 978-3-031-01587-8. DOI: 10.1007/978-3-031-01587-8_1. URL: https://doi.org/10.1007/978-3-031-01587-8_1.

[9] Franco Scarselli et al. "The Graph Neural Network Model". In: *IEEE Transactions on Neural Networks* 20.1 (2009), pp. 61–80. DOI: 10.1109/TNN.2008.2005605.

Eidesstattliche Versicherung

(Affidavit)

Name, Vorname
(surname, first name)

Matrikelnummer
(student ID number)

☐ Bachelorarbeit
(Bachelor's thesis)

☐ Masterarbeit
(Master's thesis)

Titel
(Title)

Ich versichere hiermit an Eides statt, dass ich die vorliegende Abschlussarbeit mit dem oben genannten Titel selbstständig und ohne unzulässige fremde Hilfe erbracht habe. Ich habe keine anderen als die angegebenen Quellen und Hilfsmittel benutzt sowie wörtliche und sinngemäße Zitate kenntlich gemacht. Die Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

I declare in lieu of oath that I have completed the present thesis with the above-mentioned title independently and without any unauthorized assistance. I have not used any other sources or aids than the ones listed and have documented quotations and paraphrases as such. The thesis in its current or similar version has not been submitted to an auditing institution before.

Ort, Datum
(place, date)

Unterschrift
(signature)

Belehrung:
Wer vorsätzlich gegen eine die Täuschung über Prüfungsleistungen betreffende Regelung einer Hochschulprüfungsordnung verstößt, handelt ordnungswidrig. Die Ordnungswidrigkeit kann mit einer Geldbuße von bis zu 50.000,00 € geahndet werden. Zuständige Verwaltungsbehörde für die Verfolgung und Ahndung von Ordnungswidrigkeiten ist der Kanzler/die Kanzlerin der Technischen Universität Dortmund. Im Falle eines mehrfachen oder sonstigen schwerwiegenden Täuschungsversuches kann der Prüfling zudem exmatrikuliert werden. (§ 63 Abs. 5 Hochschulgesetz - HG -).

Die Abgabe einer falschen Versicherung an Eides statt wird mit Freiheitsstrafe bis zu 3 Jahren oder mit Geldstrafe bestraft.

Die Technische Universität Dortmund wird ggf. elektronische Vergleichswerkzeuge (wie z.B. die Software „turnitin") zur Überprüfung von Ordnungswidrigkeiten in Prüfungsverfahren nutzen.

Die oben stehende Belehrung habe ich zur Kenntnis genommen:

Official notification:
Any person who intentionally breaches any regulation of university examination regulations relating to deception in examination performance is acting improperly. This offense can be punished with a fine of up to EUR 50,000.00. The competent administrative authority for the pursuit and prosecution of offenses of this type is the Chancellor of TU Dortmund University. In the case of multiple or other serious attempts at deception, the examinee can also be unenrolled, Section 63 (5) North Rhine-Westphalia Higher Education Act (*Hochschulgesetz, HG*).

The submission of a false affidavit will be punished with a prison sentence of up to three years or a fine.

As may be necessary, TU Dortmund University will make use of electronic plagiarism-prevention tools (e.g. the "turnitin" service) in order to monitor violations during the examination procedures.

I have taken note of the above official notification:*

Ort, Datum
(place, date)

Unterschrift
(signature)

***Please be aware that solely the German version of the affidavit ("Eidesstattliche Versicherung") for the Bachelor's/ Master's thesis is the official and legally binding version.**