Identifying Ultimate Beneficial Ownership from Complex Corporate Structures using Graph Machine Learning

Master's Thesis in the study course

Applied Data Science

At the NORDAKADEMIE gAG,
University of Applied Science, in
25337 Elmshorn, Germany

submitted by

Niklas Ullmann
Borgfelder Allee, 10
20537 Hamburg
MADS21o - 12055
Tel.: 0175 - 3832789

First Reviewer: Prof. Dr. Michael Schulz
Second Reviewer: Prof. Dr. Bahne Christiansen
Processing Time: 02.05.2023 - 22.09.2023

Abstract

This master's thesis addresses the complex landscape of corporate ownership structures and the identification of their Ultimate Beneficial Owners (UBOs). As businesses expand, their ownership structures become more complex, providing opportunities for concealing illegal activities such as money laundering and tax evasion. Therefore, regulatory authorities are now prioritizing the analysis of these structures in order to identify individuals with absolute decision-making power and control. Existing algorithmic approaches leverage mathematics to identify UBOs. However, these methods demonstrate poor scalability as ownership structures grow, and the convergence of the algorithms is not guaranteed.

This thesis introduces a novel approach to UBO identification, representing ownership structures as graphs and framing the UBO identification task as an inductive link prediction problem. Leveraging publicly accessible ownership data, a graph machine learning model is trained and evaluated for its overall performance and potential to address the existing problems.

The result is a machine learning model that provides reliable predictions and, in some cases, mitigates problems of the traditional algorithmic approach. These results underscore the viability of graph-based machine learning as a valuable method for identifying UBOs within complex ownership structures. A robust machine learning model for UBO identification has significant applications across various industries and scientific domains, strengthening transparency, compliance, and risk mitigation, while combating financial crime, ensuring ethical practices, and supporting responsible resource management.

Contents

List of Abbreviations

AIND	Average Incoming Node Degree
AMLD	Anti-Money Laundering Directive
BO	Beneficial Owner
BODS	Beneficial Ownership Data Standard
CNN	Convolutional Neural Network
DDS	Dataset Definition Standard
DL	Deep Learning
DSR	Design Science Research
EDA	Exploratory Data Analysis
EU	European Union
FATF	Financial Action Task Force
FN	False Negative
FP	False Positive
GAE	Graph Autoencoder
GAT	Graph Attention Network
GCN	Graph Convolutional Network
GML	Graph Machine Learning
GNN	Graph Neural Network
GraphSAGE	Graph Sampled and Aggregated
GwG	Geldwäschegesetz
ML	Machine Learning
MSE	Mean Squared Error
OO	OpenOwnership
ReLU	Rectified Linear Unit
RNN	Recurrent Neural Network
RMSE	Root Mean Squared Error
TN	True Negative
TP	True Positive
UBO	Ultimate Beneficial Owner
VRAM	Video Random Access Memory

List of Figures

List of Tables

1 Introduction

Companies are significant in our modern society, fulfilling roles as employers, service providers, and producers of consumer goods that supply our daily lives. The diversity in the corporate landscape is evident, ranging from small-scale local businesses to extensive corporations with widespread ownership.[Jam+23, cp.] In pursuing opportunities and growth, businesses naturally engage in strategies that propel them forward. Mergers, acquisitions, and strategic partnerships are common routes to expand market presence, diversify portfolios, and harness synergies. When businesses and persons hold ownership in other companies, complex ownership patterns emerge, giving rise to complex multilevel ownership structures that span industries and traverse geographical borders. These complex structures can serve legitimate purposes but can also be abused for illegal activities. For instance, they might be exploited for tax evasion, moving illicit funds, engaging in corruption, or supporting terrorist activities.[SCP20, cp.p.117ff.][She18, cp.p.10ff.]

The Panama Papers of 2016 are one of the most prominent examples of companies using complex ownership structures to conceal illegal activities—this extensive leak of documents exposed the activities of approximately 360,000 firms and individuals worldwide. The documents revealed the diverse ways companies and individuals utilized offshore and shell companies to create opaque and intricate ownership networks. These structures were employed to facilitate illegal operations such as money laundering, tax evasion, and fraud while concealing the actual individuals behind these activities.[Dom+20, cp.p.1][Zai+17, cp.]

The evolution of ownership analysis has progressed from addressing criminal cash in the 1990s to combatting terrorism financing and money laundering post-9/11, with the Financial Action Task Forces (FATFs) recommendations marking the inception of ownership transparency.[FAT23, cp.] Worldwide efforts followed, culminating in reforms reflecting the recognition that legal frameworks alone were insufficient, prompting a shift towards real-world implementation. Amid revelations of criminal wealth concealment and tax evasion, LuxLeaks in 2014 and the Panama Papers in 2016 spurred policy changes, leading to the European Union (EU) Anti-Money Laundering Directive (AMLD) and the prioritization of its enforcement.[Par15, cp.][EU 22, cp.]

Particularly interesting for the ownership analyses are the Beneficial Owner (BO), who control a company, and UBOs, who have the most significant control over a company.[And19, cp.] The current approach to determining UBOs relies on an algorithmic method with mathematical foundations that examines ownership structures to identify individuals exerting direct or indirect influence over a target company.[PPS21, cp.p.2f.] However, this approach has imperfections in practice and from a mathematical perspective. Firstly, the runtime of this method scales inadequately as the size, complexity, and depth of ownership structures increase.[PPS21, cp.p.3] Additionally, there is no guarantee that the approach will yield results; in specific scenarios, the algorithm may run indefinitely without identifying any UBOs.[MI22, cp.p.3][HL10, cp.p.163-167] [And19, cp.] These concerns raise the question of whether there are better methods for identifying UBOs within complex ownership structures.

At the same time, there has been a remarkable surge in the adoption of graph data structures and the rise of graph-based technologies. Graphs, as a data structure, have gained prominence for efficiently storing, processing, and analyzing data that has entries with numerous relationships, exceeding the capabilities of relational databases.[CMA16, cp.] This paradigm shift has found real-world applications in various domains, including social networks, navigation systems, and knowledge repositories like Wikipedia, with its 55 million interconnected articles.[Pan23, cp.] While graph data structures are well-established in industry and research, the analysis and application of machine learning on graphs is a promising effort currently under development, as the Gartner Hype Cycle for Emerging Technologies in 2023 indicates.[Lor23b, cp.][Lor23a, cp.]

This master's thesis aims to investigate the practical question of whether there is a better way to identify UBOs in complex ownership structures. The given connected nature of ownership structures of companies will be used to represent them as a graph. Current technologies of Graph Machine Learning will be applied to these ownership graphs to identify UBOs and potentially mitigate existing problems. For this purpose, a graph machine learning model will be defined according to the latest standards, trained with publicly available ownership data and evaluated concerning predictive accuracy and other metrics.

2 Research Methodology

This chapter presents methodologies to systematically investigate the prevailing challenges and formulate solutions in a scholarly manner. Its core objectives include the derivation of relevant research questions from the problems at hand, as well as the establishment of a research methodology. The chosen methodology, Design Science Research, is designed to provide a rigorous scientific framework and a pragmatic alignment with the business context, ensuring a comprehensive exploration and solving of the issues presented. This chapter thus lays the foundation for the following chapters of this master's thesis.

2.1 Derivation of the Research Question

The outcomes of the provided preliminary study, along with an examination of existing research and the following detailed literature research (Chapter 3), demonstrate an evident business demand for novel methodologies in identifying UBOs. Additionally, there is a notable absence of a research project that integrates UBO identification and graph-based Machine Learning to address the current challenges in UBO identification.[But22, cp.][MI22, cp.p.3][HL10, cp.p.163-167] Hence, the initial research question emerges:

> **Can Graph Machine Learning (GML) algorithms be applied to identify UBOs in complex corporate ownership structures?**

Subsequently, once the overall applicability is established, the succeeding question analyses the effectiveness of the Machine Learning (ML)-based approach in comparison to the conventional algorithmic method. This inquiry involves employing suitable metrics to evaluate the strengths and limitations of the chosen ML-based approach:

> **Can graph ML-based algorithms identify information equivalent to the results of the traditional algorithmic approach, and what are their strengths and weaknesses compared to the latter?**

The literature and practical observations suggest that employing an ML-based approach can offer multiple advantages over algorithmic approaches. For instance, it can exhibit greater flexibility in recognizing complex patterns within datasets.[Bis06, cp.p.267, 654] It can adapt to dynamic changes, potentially enhancing the detection of fraudulent or illegal activities.[Con+22, cp.] The two research questions can be summarised as follows:

> **How can GML algorithms be applied to complex corporate ownership structures to identify UBOs, and what advantages and disadvantages do they offer compared to established algorithmic approaches with respect to accuracy and performance?**

2.2 Design Science Research

In order to answer the proposed research question adequately within the scientific framework of this master's thesis, the following section presents Design Science Research (DSR) as the selected research methodology and describes the methodological procedure.

The DSR paradigm is a problem-solving paradigm. By producing new artefacts and generating knowledge through innovative solutions to real-world problems, DSR aims to advance human knowledge.[vBHM20, cp.p.1-3]

The DSR framework consists of three conceptual domains, as shown in Figure 1. The environment consists of people, organizations, and technologies that define an existing need or problem. On the other hand, the knowledge base provides the scientific fundamentals for fulfilling the need; these can be scientific knowledge or methods. Both domains are merged in the centered research block, where theories or artefacts are developed, implemented and validated for the described problem.[Hev+04, cp.p.80]

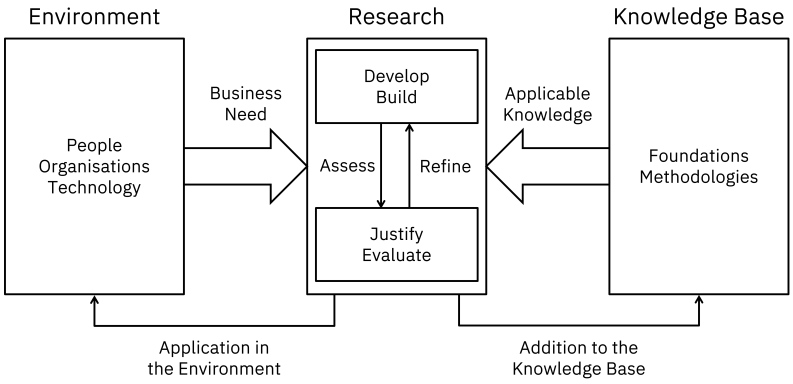

Figure 1: DSR Framework [Hev+04, adapted]

The generated artefact and knowledge from the research domain can solve the initial problem and contribute to the knowledge base. Depending on the general state of research and the state of the problem, the type of resulting artefact for the environment can range from invention, improvement, adaption or routine design [VK15, cp.]. Especially in IT, these can be concepts, models and implemented prototypes [Öst+10, cp.p.4]. The contribution to the knowledge base can be either the theory behind solving the problem or the realization that the problem cannot be solved as expected with the available knowledge and methods [GH13, cp.p.341f.].

Resulting from the general DSR framework, the DSR method process model, shown in Figure 2, has been established by Peffers et al. [Pef+07], which is also the guideline for the execution and elaboration of this master's thesis. The DSR process begins with the problem identification and the determination of the business need. In their research, Polovnikov, Pospelov, and Skougarevskiy [PPS21] as well as Magnanimi and Iezzi [MI22] show that the current approach to determining UBOs is subject to several problems. However, there is a growing interest in analysing and understanding ownership structures in the business community, especially from the regulatory authorities' perspective. The second step after the problem identification refines the problem and evaluates the existing scientific knowledge and methods.[Pef+07, cp.p.55] The assessment of current methodologies and existing knowledge is presented in Chapter 3 as part of the literature review.

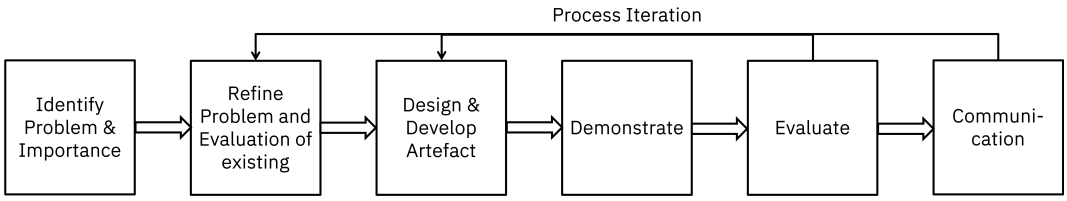

Figure 2: DSR Process [Pef+07, adapted]

During the design and development, the artefact is derived using recognized methods, justified as far as possible, and distinguished from known solutions in science and practice [Öst+10, cp.p.5-4]. In the demonstration phase, the artefact provides evidence and showcases its efficacy in resolving one or more instances of the problem. This can be accomplished through experimentation, simulation, case study, proof, or other appropriate activities.[Pef+07, cp.p.55] The artefact of this thesis is developed and demonstrated in Chapter 4.

After a successful demonstration, rigour requires checking the created artefact against the initially defined goals and metrics in the evaluation phase. Evaluation involves assessing the extent to which the artefact effectively addresses the problem. This evaluation compares the solution's objectives with the observed outcomes resulting from the artefact's utilization in the demonstration. Any suitable empirical qualitative or quantitative evidence or logical proof can be considered in the evaluation [Pef+07, cp.p.55]. The developed artefact is evaluated in Chapter 5.

The last step of the DSR process is the communication of the research results [Pef+07, cp.p.56]. As pointed out in the description of the general DSR framework, the result can contribute to business and scientific knowledge. The communication of results is done after the completion of the master's thesis and, therefore, is a scientific contribution and addition to the business context.

The steps of the DSR process model are performed in a consecutive sequence. However, if the evaluation or communication phase indicates potential improvements or more profound adjustments to the business need, several steps can be taken back in the process to enhance the artefact further.

3 Literature Review

As the second step in the DSR process, the following chapter addresses the raised problems and supports them with a literature review. This literature review aims to deepen the understanding of the business problem and expand the knowledge about the current state of research and domain to refine the given problem further and to collect solution approaches for developing the artefact. For this purpose, the legal basis of UBOs is first examined in more detail, and the problems of the existing approach are highlighted. Next, graphs are considered as a data structure, including their unique features and the extent to which machine learning can be applied to graphs.

3.1 Corporate Ownership

Businesses are the building blocks of every modern economic society. They significantly influence everyday life as they provide consumer products, employment, services, taxes for the state and much more.[Jam+23, cp.]

Business ownership takes on different forms, each with distinct characteristics and implications. In a sole proprietorship, a single individual takes the reins of the business. This grants them absolute control but also translates to total liability. In essence, the owner and the business are one entity. Any financial losses, debts, or legal issues incurred by the business directly impact the owner's assets. This structure often suits small local businesses where a sole owner can handle all facets efficiently.[Rob11, cp.p.9f.] Partnership as a business involves collaboration, with two or more individuals sharing ownership. Each partner brings unique skills, enriching the business's capabilities. The liability and risks, however, are shared among partners. This structure mitigates the burden on any single individual in case of financial downturns or legal matters. Partnerships are ideal for small to medium-sized enterprises where a collective approach to decision-making and resource pooling is advantageous.[Fel06, cp.p.341] As businesses grow, incorporating as a separate legal entity becomes appealing. A corporation is distinct from its owners and can enter contracts, own property, and be held responsible for its actions. Shareholders own the corporation through shares of stock, allowing for distributed ownership. A board of directors oversees corporate governance, ensuring alignment with goals. The benefits of a

corporation include limited liability, wherein shareholders' assets are protected, and access to substantial financial resources. Corporations also offer continuity beyond the involvement of original owners, making them suitable for long-term operations.[SCP20, cp.p.117ff.]

Mergers and acquisitions are pivotal strategies in the corporate world, employed to achieve strategic objectives and elevate companies' competitive standing. While they share certain common traits, they also possess distinct characteristics that set them apart.[FM97, cp.p.40] In both mergers and acquisitions, the fundamental objective is integrating operations, assets, and personnel for improved efficiency and value creation. However, the nature of these actions varies considerably. Mergers entail the fusion of two relatively equal-sized entities into a new unified organization, which conveys a sense of partnership and equality between the merging companies. On the other hand, acquisitions involve one company assuming control over another, leading to a clear hierarchy between the acquiring and target companies. This distinction is reflected in ownership distribution and management authority.[Wri+02, cp.p.41-44] Another facet of differentiation lies in the purpose and strategy underlying mergers and acquisitions. Mergers often materialize when companies foresee the potential for synergies that will amplify their overall value and competitive prowess. In contrast, acquisitions are frequently pursued to access specific assets, technologies, markets, or resources the target company possesses.[Bru04, cp.p.4-8] Ultimately, whether through mergers or acquisitions, companies strategically navigate these transactions to realize growth, innovation, and enhanced market presence.[She18, cp.p.10ff.]

In business and corporate governance, an ownership structure refers to the arrangement and distribution of ownership rights, shares, and control among individuals, entities, or shareholders within a company or a group of related companies and natural persons. It outlines the hierarchical relationships, decision-making authority, and levels of influence held by various stakeholders, including shareholders, executives, management teams, and external investors.[DC18, cp.p.86-88] Ownership structures can be complex and dynamic, shaped by mergers, acquisitions, joint ventures, and investment activities. They significantly determine how power, responsibilities, and economic benefits are allocated within an organization or across interconnected entities. Complex ownership structures involving multiple companies and legal entities emerge from acquisitions and other ownership transactions like building subsidiaries and divisions, using holding structures or assembling joint ventures and alliances.[LL08, cp.p.9-11] Such complex ownership structures enable companies

to strategically expand their reach, manage risk, and capitalize on synergies in today's dynamic business landscape.[Atz+20, cp.p.556]

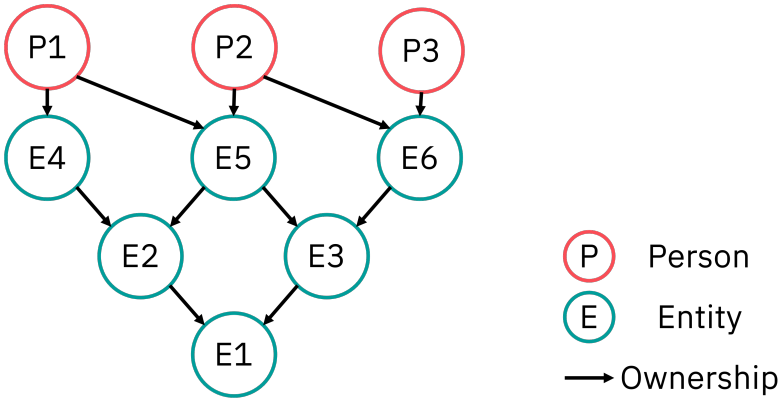

Figure 3: Ownership Structure Example

Such a complex ownership structure could look like the one displayed in Figure 3. Each node represents either an entity (E) or a person (P). The term entity is used as a generic term for companies and all other corporate types. Arrows between nodes indicate a type of ownership. The target entity E1 is owned by two other entities (E2 and E3). The ownership structure continues upwards on three levels. The end of the ownership structure is always natural persons, as others cannot possess them.

In legal requirements, the disclosure of complex ownership structures towards financial authorities serves a dual purpose. While such complex structures are not inherently illegal, they often become conduits for covert criminal activities and fraudulent practices. These complex ownership arrangements can be exploited for illicit purposes, including money laundering, tax evasion, terrorist financing, and fraud.[FAT18, cp.p.25-30] Recognizing potential risks, numerous countries have established guidelines to shed light on these ownership structures. International organizations such as the FATF and legislative bodies like the EU through the Fourth and Fifth AMLDs emphasize the need for accurate and timely information on BOs (See Chapter 3.1.1). „The need for accurate and up-to-date information on the BO is a key factor in tracing criminals who might otherwise hide their identity behind a corporate structure."[Par15, §14] Many nations within and outside the EU have adopted similar measures to combat money laundering. Notably, there is a significant interest in comprehending ownership structures and the entities or individuals wielding the most influence — the BOs. This collective effort reflects the commitment to enhance transparency and curb criminal financial activities.

3.1.1 Beneficial Ownership

The FATF, the EU, and its member states, which must implement the EU directives individually, all have different definitions of a BO. However, it is clear from the definitions that a BO is a natural person or a legal entity that owns and can influence another entity to a certain percentage. The respective authorities decide on this percentage themselves [T-r19, cp.p.2-3]. According to a study by United Nations Office on Drugs and Crime [Uni22], which examined the definitions and implementations of BO rules of 38 countries, the threshold is often between 10% and 25%. Owners below this threshold do not have to disclose their ownership to the state. An owner above this threshold is considered to impact the target entity significantly [Uni22, cp.p.5]. In order to establish a standard definition for this scientific work, the German definition of a BO will be used in the following.

According to the definition of the German Money Laundering Act (Geldwäschegesetz (GwG)), any natural person or legal entity who:

1. holds more than 25% of the capital shares, or
2. controls more than 25% of the voting rights, or
3. comparably exercises control

are considered BOs of a target entity [Jus17, cp.§3 (translated)].

Ownership in a company is characterized by two fundamental types of rights: capital shares and voting rights. Capital shares are a share in the company's profitability, encompassing dividends, liquidation proceeds, and sales. Voting rights, on the other hand, grant the authority to influence significant company decisions. The potency of a shareholder's voting power is not solely determined by their capital shares ownership percentage but is significantly impacted by the distribution of other shares. Typically, there is no requirement for an equal distribution of capital or voting rights among shares.[T-r19, cp.p.2-3] In addition to these main ways an owner can influence the company, there are many more and more complex ways, which are collected under point 3 of the enumeration.

As shown in Figure 3, BOs can directly influence the companies they own and indirectly influence them through the exercise of power through an ownership structure. For example, Person 1, 2 and 3 all influence Entity E1, even though they do not directly own shares in the entity. Understanding direct and indirect influence in ownership structures is critical in determining BOs.

Direct Control

Direct ownership refers to the straightforward ownership of assets or shares by an individual, entity, or organization. Ownership is explicit and unmediated, as the owner possesses legal and financial rights over the assets without intermediaries. Direct ownership gives individuals and entities a high degree of control over their assets.[PPS21, cp.p.3]

Indirect Control

Calculating indirect ownership is more complex than direct ownership due to the complex relationships that can emerge within ownership structures. Direct ownership involves straightforward connections between shareholders and the entities they own shares of. However, the task becomes progressively more complicated as ownership extends through multiple levels and involves various entities. In indirect ownership, ownership chains can span numerous levels of subsidiaries, holding companies, and interconnected entities. Determining the cumulative ownership percentages at each level requires accurately tracing ownership paths, accounting for potential overlaps, cross-ownership, and interlocking directorates.[MI22, cp.p.3]

$$\sum_{k=1}^{\infty} A^k = A^1 + A^2 + A^3 + \cdots + A^\infty \tag{1}$$

It is widely acknowledged that direct connections alone are insufficient for uncovering obscured participants who hold influence over entities. Thus, consideration must be given to the indirect impact that a participant, denoted as i might exert on organization k through interactions with all other participants within the network. In the context of control by transitivity, the extent of indirect control is proportionate to the net influence originating from i and directed toward k, given the network's interconnections.[VGB11, cp.p.1] This can be formally expressed as the summation of the powers of the weighted adjacency matrix A, shown in Equation 1. The concept of the adjacency matrix is explained in detail in Chapter 3.2.1. The first term (A^1) represents the contribution from direct control, followed by contributions from paths of length 2, length 3, and so forth.[PPS21, cp.p.3]

The summation process extends from the value of 1 onward until reaching positive infinity. As this summation progresses, it eventually converges to a static matrix result, signifying that further iterations do not alter the outcome.

The resulting matrix allows indirect ownership to be inferred. Similar to how direct ownership was deduced from the ownership matrix, as discussed earlier, information about indirect ownership can be taken from the resulting matrix.[PPS21, cp.p.3]

Ultimate Beneficial Ownership

Beneficial Owners are essential for regulatory authorities to understand the underlying ownership structures of an entity. However, determining the UBO in an ownership network is equally important to promote transparency, regulatory compliance, risk mitigation, and informed decision-making. It helps curb illicit activities and ensures the integrity of business operations, making it an essential practice for a responsible and well-functioning business environment.[Lex23, cp.]

The FATF defines an UBO as: „The natural person(s) who ultimately owns or controls a legal entity [...] Only a natural person can be an Ultimate Beneficial Owner, and more than one natural person can be the UBO of a given legal entity or arrangement“[FR22, p.5]

The identification and calculation of UBOs within a company's ownership structure can be achieved through defined calculation methods. However, it is essential to recognize that power and control can be exerted on a company in various ways, whether directly or indirectly. UBOs can be derived by applying the German definitions of a BO and assessing the different control types.[T-r19, cp.] This involves separate consideration of capital shares, voting rights, and other forms of power. The identification process involves creating adjacency matrices that contain the weighted relationships between all participants. Control calculation is carried out using a formulated method. Each type of control is examined individually to adhere to its hierarchy as per legal requirements. The individual with the largest capital shares is identified as the UBO by the specified guidelines. In cases where multiple individuals possess equal shares on one level, the distribution of shares on the underlying level is used as a decision criterion. This approach ensures a transparent allocation and consideration of the hierarchy of control shares, enabling precise identification of UBOs in a complex corporate structure.[PPS21, cp.p.3]

3.1.2 Limitations of the Calculation

Calculating ultimate ownership using the sum of matrix powers of the adjacency matrix can encounter several problems, depending on the given ownership structure, resulting in a long calculation time or no convergence of the algorithm.

Time Complexity

Understanding and considering an algorithm's time complexity is essential in computer science research and application. Time complexity refers to how the computational time required by an algorithm grows with the size of the input data.[DK00, cp.]

Time complexity analysis enables assessing the efficiency of different algorithms and making informed decisions on algorithm selection. Knowing how an algorithm's runtime scales with increasing input size allows the differentiated and justified choice of the most efficient algorithm to solve a particular problem. This consideration becomes especially critical when dealing with large datasets or computationally intensive tasks. Moreover, time complexity analysis allows to compare different algorithms that tackle the same problem. Computer scientists can quickly grasp an algorithm's relative efficiencies and trade-offs by categorising them into complexity classes. This enables an informed decision on the most suitable approach for a given task.[OB17, cp.p.1758]

The Big$\mathcal{O}$ notation, a mathematical notation representing the upper bound of an algorithm's time complexity, plays a central role in time complexity analysis. It abstracts away the details of an algorithm's execution time and focuses on the overall growth pattern as the input size increases. In Big$\mathcal{O}$ notation, the efficiency classes describe the growth rates of functions and help categorise algorithms based on their time complexity.[MA22, cp.p.1f.] Standard efficiency classes of the Big$\mathcal{O}$ notation are:[DSR11, cp.p.845f.]

- $\mathcal{O}(1)$ - Constant Time: Algorithms with constant time complexity have a fixed execution time that does not depend on the input size. For example, when determining if a binary variable is odd or even.
- $\mathcal{O}(logn)$ - Logarithmic Time: An example of an algorithm with logarithmic time complexity is a binary search.
- $\mathcal{O}(n)$ - Linear Time: These algorithms scale linearly with the input size, for example, iterating over a list searching for an item.

- $\mathcal{O}(n^x)$ - Polynomial Time: The algorithms from this efficiency class have running times that grow as a power of the input size. Typical exponents are 2 , 3 or 4. For example, simple sorting algorithms like bubble or selection sort run in quadratic time.
- Other classes are $\mathcal{O}(2^n)$, with exponential growth of time, and $\mathcal{O}(n!)$, where running times grow factorially with the input size.

Depending on the problem, different classes are considered efficient enough, considering time and hardware effort. Figure 4 shows the big runtime difference of the different classes in dependence on n again. Algorithms with a runtime from the green zone are generally fast. The yellow zone already describes a moderately good runtime, and runtimes from the red zone are considered slow.[Hei21, cp.][Sam23, cp.]

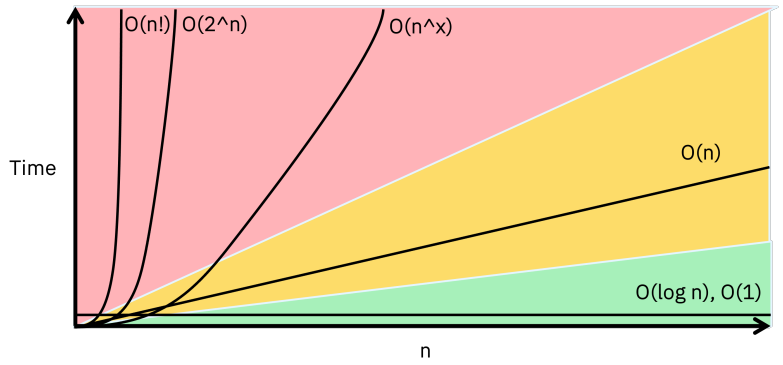

Figure 4: Time Complexity Comparison [Sam23, adapted]

In order to conduct a comprehensive analysis of the computational efficiency of the algorithm used to determine UBO in a complex ownership graph, the algorithm is classified based on its efficiency. This evaluation considers the dimensions of the input data and the number of matrix powers required until the summed matrix converges.

$$n(k) = \sum_{i=0}^{k}(m^i) = \frac{m^{k+1} - 1}{m - 1} \tag{2}$$

The classical matrix multiplication of an $n \times n$ matrix is known to have a time complexity of n^3 due to the involvement of numerous scalar multiplications and additions [Str69, cp.p.354]. Whereby n denotes the dimensions of the adjacency matrix necessary for the algorithmic calculation of the UBOs and is equal to the number of nodes in an ownership structure. Dwibedy and Mohanty [DM22] estimate the number of nodes in a perfect tree network using the Equation 2,

where they assume the tree network to have an equal number of child nodes (parent companies). Where m is the average number of parent companies a company has, and k is the number of levels in the considered ownership structure. This results in an exponential increase of nodes in an ownership network with increasing depth.[DM22, cp.p.12]

To assess the evolution of necessary matrix multiplications, a tabulation in Appendix Table 3 records the number of matrix multiplications required to compute the formula up to a fixed value k. Remarkably, the last column of the table indicates that the necessary matrix multiplications for the sum of the matrix power up to k follow the pattern of the triangular number series [Spi08, cp.p.20-22]. Consequently, the Big$\mathcal{O}$ notation for calculating the UBO can be derived by the following Equation 3.

$$\mathcal{O}(n^3 \cdot \left(\frac{(k-1) \cdot k}{2}\right)) \tag{3}$$

Various mathematical tricks and numerical optimizations can further enhance the algorithm's performance. For example, the complexity of the individual matrix multiplications can be reduced by the work of Williams et al. [Wil+23] from a complexity of n^3 to a complexity of $n^{2.371}$ and thus approaches more of a quadratic increase of the runtime instead the cubic. Also, the algorithm's required matrix multiplications can be reduced to $log_2(k)$ by binary exponentiation [Knu97, cp.p.461-463][Möl11, cp.p.84f.]. This results in the following classification of the optimized algorithm in the Big$\mathcal{O}$ notation:

$$\mathcal{O}(n^{2.371} \cdot log_2(k)) \tag{4}$$

However, with this estimation, the described algorithm still falls into the category of algorithms whose runtime increases polynomially with the data size and is thus one of the worse algorithms from the point of view of performance.

Non-Convergence

Moreover, it is more complicated to determine how many matrix powers k have to be summed up until the sum converges and the UBOs can be deduced from the resulting matrix. Ideally, k is equal to the maximum depth of the ownership structure between the target entity and the most distant person. This follows from the fact that $a^k_{(i,j)}$ is always the power that a node i exerts on an entity j in a distance of k[MI22, cp.p.3].

However, this ideal case cannot be met in a mathematical and a domain edge cases, so there is no convergence of the sum, and the UBOs cannot be calculated with the described method.

From a mathematical point of view, the series in Equation 1 is divergent (non-convergent) for matrices with the spectral radius $\rho(A) \geq 1$ or, more simply, when any eigenvalue of the matrix is ≥ 1 [Bon+12, cp.p.74].

When dealing with matrices, particularly the sum of matrix powers from 1 to k, their convergence behaviour highly depends on their eigenvalues. In this context, the highest eigenvalue holds a special significance. Eigenvalues represent how much a matrix stretches or compresses vectors in a particular direction. For convergence to occur, the matrix must be stable, meaning that its influence should diminish over successive powers, eventually leading to a steady state. In mathematical terms, this means that the magnitude of all eigenvalues should be less than 1. When the highest eigenvalue is greater than or equal to 1, it implies that the matrix amplifies at least one direction rather than reducing its influence. As a result, when summing up the powers of this matrix, the effect of the higher eigenvalue keeps growing with each power iteration, preventing the sequence from converging to a stable value.[HL10, cp.p.163-167]

Additionally, a particular ownership structure causes problems for the present algorithm. Circular ownership is a distinct approach to structuring business entity ownership. In this arrangement, two or more entities mutually own each other, creating a circular pattern within the ownership structure. An individual may appear to hold a minor ownership percentage that does not trigger disclosure requirements, yet this seemingly low ownership conceals actual control. Companies can establish circular ownership structures to obscure actual ownership and control using intricate ownership loops.

These structures can take various forms and bring with them various problems. Open-circle structures (like in Figure 5) involve external owners who appear to hold small percentages but may, in reality, possess a substantial stake. These setups exploit regulatory thresholds since the ownership of P1 must not be disclosed, allowing ultimate owners to remain hidden. On the other hand, circular ownerships serve legitimate purposes, such as unintentional evolution through mergers or acquisitions and tax efficiency. Nevertheless, it can also be exploited for illicit reasons like tax evasion, money laundering, and sanctions avoidance.[And19, cp.]

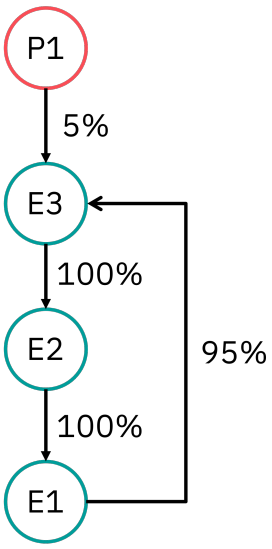

Figure 5: UBO Hiding

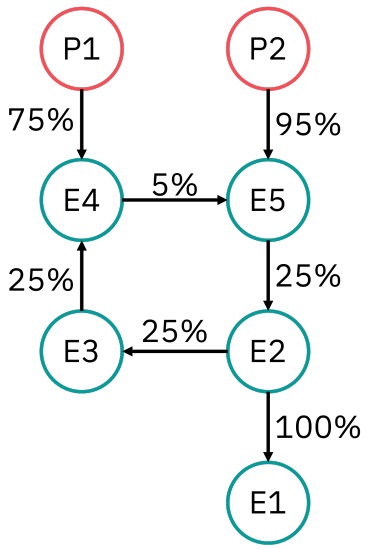

Figure 6: Non-Convergence

Circular ownership structures also pose challenges for the calculation algorithm. Based on the example in Figure 6, the calculation over the matrix powers would never converge. With each loop, with each new matrix power, through the ownership circle, the influence of P1 and P2 changes until the changes become infinitely small. On paper, the algorithm would never converge. However, every computer is limited by the maximum precision of the floating point variable. At some point in the summation of the matrix powers, a point is reached where the change becomes too small and the variable no longer changes, thus reaching „convergence“. For the relatively flat ownership structure in Figure 6 and a used precision of floatingpoint64, enabling up to 16 decimal places to be displayed [DM08, cp.p.8], the algorithm needs the sum of A^1 up to A^{39} to reach convergence.

The most common type of cyclic ownership is corporate cross-ownership. In this strategic arrangement, two or more companies hold ownership stakes in each other, creating a network of interconnected ownership relationships. This strategic manoeuvre is often driven by pursuing synergies, risk diversification, or controlling influence over partner companies. However, cross-ownership can raise concerns about potential conflicts of interest, anti-competitive behaviour, and transparency issues.[HL23, cp.p.3f.]

Exploiting these structures allows ultimate owners to remain obscured, evading due diligence requirements and sanctions detection by hiding ultimate owners or complicating the UBO determination. The legality of circular ownership varies across jurisdictions. It is common and legal in Russia, permissible in certain conditions and at low share percentages in Italy, the Netherlands, and

South Korea, but illegal in the UK and Norway, among other places and therefore a problem that needs further consideration during the design of the artefact.[Ste20, cp.]

3.2 Graph Machine Learning

Graph machine learning refers to the application of machine learning techniques specifically designed for analyzing and processing graph-structured data. It involves mapping the features of a graph to feature vectors in an embedding space. The primary goal of graph machine learning is to extract relevant features from the graph, enabling downstream tasks such as node classification, link prediction or graph classification. By harnessing the inherent structure of graphs, graph machine learning provides a powerful and meaningful approach to graph analysis.[Xia+21, cp.p.1][Zho+20, cp.p.1]

However, the non-Euclidean nature of graphs poses particular challenges to processing them, as commonly used algorithms from Deep Learning (DL), such as Convolutional Neural Network (CNN) or Recurrent Neural Network (RNN), only work on data from a Euclidean space and cannot be directly applied to graphs [Bro+17, cp.p.1][Lia+22, cp.p.1].

The following section is intended to take a closer look at graphs as a data structure, to show possible fields of application of graph machine learning and to explain state-of-the-art methods of graph machine learning for later use.

3.2.1 Graph Datastructure

Generally, a graph can be any mathematical object involving points and connections between them. Formally a graph G is defined as a tuple of two sets, a non-empty finite set V and a finite set E of unordered pairs of distinct elements of V, as shown in Equation 5. The elements of V are called vertices or nodes, and the E elements are called edges. Each edge connects one or two vertices, referred to as its endpoints. The purpose of an edge is to link these endpoints in the graph.[Wil09, cp.p.17]

$$G = (V, E) \tag{5}$$

Attributes can be added to nodes and edges to enrich graphs with further information and apply them to real problems. Mathematically, a vertex attribute is defined as a function from V to some set of possible attribute values. Likewise, an edge attribute is a function from E to some possible attribute values. Typical attributes are, for example, the node type or the direction of the edge.[GYZ17, cp.]

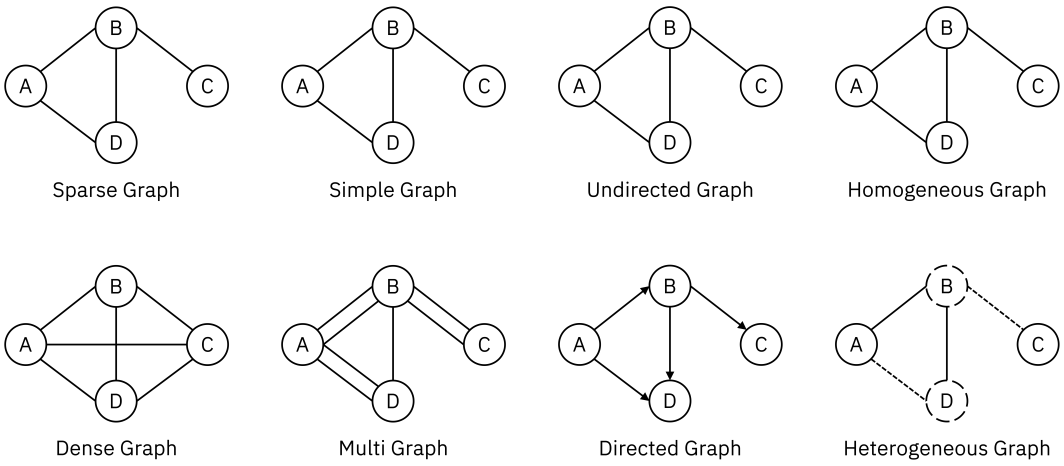

Figure 7: Subcategories of Graphs

Each graph can be further specified into subcategories depending on its structure and given node and edge attributes. Graphs can be categorised by the number of existing edges in a graph, where a graph is called sparse if the number of edges between n nodes is smaller or equal than n [Pre99, cp.p.543]. Conversely, a graph with close to the maximum number of edges is considered dense [KB16, cp.p.269], as can be seen in the left column of Figure 7. Graphs where the vertices are connected with only one edge, are called simple graphs. Moreover, graphs are called multigraphs when two vertices are connected with multiple edges [Tru93, cp.p.123], as can be seen in the second column from the left in the Figure 7. Additionally, graphs can be directed or undirected. In a directed graph, the edges are defined as an ordered pair of two vertices, where the order of the vertices indicates the direction of the edge. In an undirected graph, the pure order of the vertices in an edge is indifferent [Die17, cp.p.28]. The previous classifications and representations of graphs assume homogeneity of nodes and edges, meaning there is always only one type of node and one type of edge in a graph. This exists, for example, in simplified social networks where there are only people as nodes and friendships as edges.[WF94, cp.] However, modelling more complex and real-world data structures requires introducing different types of nodes and edges to model the data more accurately. Graphs with

multiple node and edge types are called heterogeneous graphs.[HJ23, cp.] As seen in the top row of Figure 7, the same graph can carry multiple properties simultaneously. Properties from the upper and lower row can also be mixed as long as no directly subordinate properties are chosen since they are always opposite.

Graph Representations

Efficient and reliable storage of graphs, particularly the representation of graph edges, plays a crucial role in various scientific domains and real-world applications since real-world examples of graphs can scale up to millions of nodes and even more edges between them. Efficient storage of graph edges is crucial for scalable processing of graph data and enabling efficient operations on the graph structure. Graph algorithms heavily rely on edge traversals, and the storage format impacts the efficiency of accessing and manipulating edges during various graph operations. Optimized storage schemes reduce computational resources and enable faster and more scalable graph analysis. In memory-constrained scenarios, efficient storage becomes even more critical, allowing for better resource utilization and handling large graphs within limited environments.[SS12, cp.p.179]

Graphs can be represented using different storage formats, including adjacency matrices, adjacency lists and edge lists.

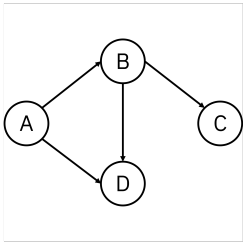

Figure 8: Example Graph

	A	B	C	D
A	0	1	0	1
B	0	0	1	1
C	0	0	0	0
D	0	0	0	0

Figure 9: Adjacency Matrix Example

An adjacency matrix is a commonly used data structure for representing the structure of a graph and analyzing its properties. It is an $n \times n$ boolean matrix, where the entry $adj[i, j] = 1$ if there is an edge from node i to node j, and $adj[i, j] = 0$ if no such edge exists. It allows for the representation of self-edges (edges from a node to itself), but it does not handle multiple edges between distinct nodes. Changing the boolean matrix to an integer or float matrix, edge weights can be stored in the adjacency matrix for weighted graphs or the number of edges between nodes for a multigraph.[Nel23, cp.] Adjacency

matrices offer convenient and efficient operations, allowing for constant-time addition, removal, and edge existence checks. Adjacency matrices are optimal for dense graphs, where the number of edges is comparable to the square of the number of vertices. In such cases, adjacency matrices provide a memory-efficient representation without unnecessary space overhead. Sparse graphs with considerably fewer edges than the square of the number of vertices can result in wasted space and inefficient memory usage with adjacency matrices.[SS12, cp.p.182] Figure 9 shows the adjacency matrix for the graph of Figure 8.

The adjacency list is a representation of edges in graph theory. Figure 10 represents the adjacency list for the graph of Figure 8. In an undirected graph, each node gets its list of adjacent nodes. For each node v, a list of the edges (v, u) is stored, where v is the source and u, the destination, is another node in the graph G. Depending on the intended graph structure, the adjacency list can be formulated as a multiset for multigraphs, a set for directed graphs or others.[GT02, cp.p.299f.][La 21, cp.] The adjacency list stores edges in a more compact form compared to the adjacency matrix, but with the disadvantage that adding and removing and the existence check of an edge between two nodes takes longer.[SS12, cp.p.182]

Node	Adj. List
A	[B, D]
B	[C,D]
C	[]
D	[]

Figure 10: Adjacency List Example

Edge List
[[A,B], [A,D], [B,C], [B, D]]

Figure 11: Edge List Example

An edge list, in contrast, is a single list containing two tuples of a starting node and the end node of every edge of the graph. Figure 11 shows the edge list for the graph of Figure 8. In an ordered graph, the order of the tuple items dictates the edge direction; otherwise, the order can be ignored.[SS12, cp.p.181] The edge list is memory efficient with some disadvantages when it comes to editing and querying.[GT02, cp.p.298]

Non-Euclidean Space of Graphs

As mentioned at the beginning, graphs are located in non-Euclidean space and thus pose unique requirements for analysis and learning methods, which are not

covered by conventional methods, such as those from image, text, or tabular data processing.

In Euclidean space, denoted by $\mathbb{R}^n$, data can be represented as points in an n-dimensional linear space. For instance, the x and y coordinates define the pixel locations in image representation, while the z coordinate represents the intensity in grayscale images. Consequently, images can be viewed as data in a three-dimensional Euclidean space.[Lia+22, cp.p.3] Besides images, texts can be displayed in one-dimensional space and tabular data in n-dimensional space. Euclidean space-based machine learning models rely heavily on fixed-sized neighbourhoods and distance-based measurements to learn from given data.[Zho+20, cp.p.1]

Due to their dynamic dimensions and relations, graphs cannot effectively be encoded in Euclidean space. Mapping such data to Euclidean space would lose essential information, particularly the relationships between data entries.[Lia+22, cp.p.3] In non-Euclidean data, the absence of familiar properties such as a standard system of coordinates, vector space structure, or shift-invariance becomes apparent.[Bro+17, cp.p.1] Graph-structured data presents challenges in defining networks with consistent structural priors, as the structures can be arbitrary and vary significantly across different graphs and even within nodes of the same graph. Unlike regular image domains, where convolutions can be applied uniformly due to the consistent neighbourhood structure of pixels, convolutions cannot be directly applied to irregular graph domains. This is because each node in a graph can have a unique neighbourhood structure, making it impossible to establish a node ordering. Additionally, Euclidean convolutions heavily rely on geometric priors like shift invariance, which do not extend to non-Euclidean domains where translations may not even be defined.[Cha+22, cp.p.2]

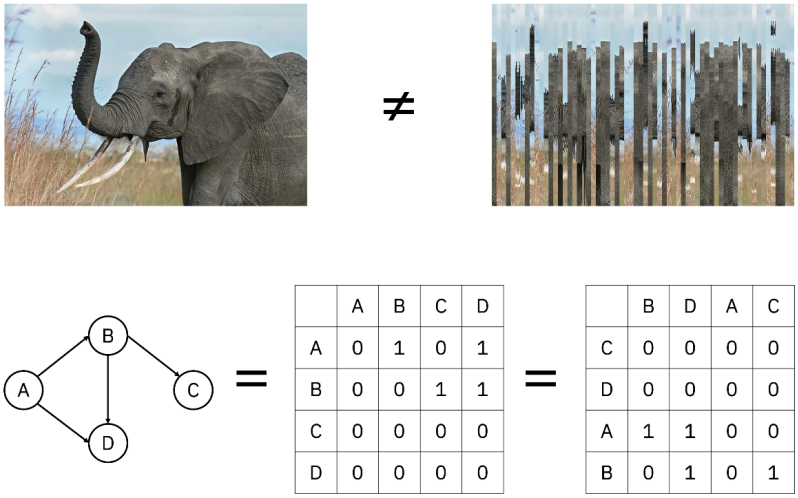

Figure 12: Permutation Invariance and Variance [Fou23, adapted][Muh11]

The most significant difference between data in Euclidean and non-Euclidean space is the behaviour of the models when permutations occur in the representations of the data. Since graphs have no logical order, a graph machine learning model should be permutation invariant. Representations of a graph, such as an adjacency matrix, can differ depending on the starting point but still display the same graph. Therefore, two different permutated adjacency matrices can describe the same graphs, as shown in the lower part of Figure 12. In contrast, the image of the elephant significantly changes when it is column-wise permuted, as shown in the top part of Figure 12. Accordingly, this permutation invariance of graphs must also be mapped by graph machine learning models since there is never a defined correct input sequence of nodes and edges, as is the case for images or texts. This behaviour is called permutation invariance and is defined as displayed in Equation 6. [Fou23, cp.]

$$f(P(G)) = f(G) \tag{6}$$

Where G is the graph, $P(G)$ is the permutated graph, and f is the model. It shows that the output from a permutation invariant model is the same whether permuted by the original graph or not. Machine learning models from Euclidean space are not permutation invariant, as the following example shows, since permutation shifts distances and neighbourhoods, producing a different result. Which makes them unsuitable for being applied to graphs.[Zho+20, cp.p.58]

Applications of Graph Datastructures

Graphs as a data structure provide an excellent way to store, retrieve, and analyze data meaningfully and effectively in many real-world problems.

When considering why to choose a graph data model over other options like relational or object-oriented models, the answer lies in the focus on interconnectivity and topology. Graph models excel in managing data where relationships between data units are as crucial as individual data. Graphs offer simplicity, naturalness, and intuition as knowledge representation systems, surpassing the limitations of linear writing formats. Graph data structures allow information about entities to be consolidated within a single node, and connections between nodes depict related information. Graph-based queries directly address the inherent graph structure, enabling high-level abstraction and specific graph operations like finding shortest paths or determining subgraphs. Moreover, graph databases leverage specialized storage structures and efficient algorithms to optimize the implementation of graph operations on the data.[FHL18, cp.p.5f]

Graphs have numerous real-world applications, making them a crucial data structure to understand. Social networks utilize graph algorithms to suggest user connections. Google Maps employs path-finding algorithms to find the shortest routes between locations. Recommendation engines use graph-based analysis to make personalized recommendations based on user interactions. Additionally, graphs find applications in computer vision, natural language processing, bioinformatics, and artificial intelligence, enabling the representation and analysis of complex relationships.[Pan23, cp.]

3.2.2 Goals and Applications

Possible applications and problems on graph data, which graph machine learning methods can solve, are manifold. However, a general distinction is made between three types of GML tasks: node, edge or graph level tasks.[San+21, cp.]

In node-level tasks, an entire graph serves as input to assign a class to each node or predict a node attribute. For example, individual user's interests can be predicted based on a social network. Edge-level tasks focus on the prediction of connections between two nodes and also the prediction of edge attributes. For example, new matching friendships can be predicted in a social network. level tasks are concerned with the prediction of individual subgraphs. This can be, for example, a classification of molecules, whether they respond to a particular disease or not.[Zha+21, cp.p.6-8]

Moreover, graph machine learning tasks can be divided by the need to generalize their learned knowledge on new graphs. Transductive and inductive graph machine learning approaches are two different ways of modelling and learning from graph data. In transductive settings, the focus is on inferring information about or between observed nodes in a fixed graph, as shown on the left side of Figure 13. These methods assume that all nodes in the graph are observed during training. They can be used to predict labels for all nodes or make predictions based on partial labelling. However, transductive models have limitations when it comes to generalizing to new nodes or new graph instances.[YCS16, cp.p.4f.]

Figure 13: Transductive vs. Inductive GML [GBH22, adapted]

Inductive settings aim to train models that can generalize to unseen nodes, edges, or graphs not observed during training, as seen on the right side of Figure 13. The goal is to learn embeddings that can be applied to new and unseen test graphs. Inductive learning allows for generalization to new instances and is particularly useful for tasks involving dynamic graphs. It enables models to make predictions or embed new graphs based on previously learned patterns. While transductive learning leverages unlabeled test data during training, inductive learning is desired for tasks that require generalization to unobserved instances or larger-scale applications.[Cha+22, cp.p.8]

$$f : G \rightarrow \mathbb{R}^n \tag{7}$$

In order to make the described downstream tasks possible at all, variables and information are required in a Euclidean space. The goal of any graph machine learning algorithm is primarily to learn a function f, as shown in Equation 7, to map the complex non-Euclidean structures of a graph into the continuous Euclidean space, such that essential graph properties (e.g. local or global structure) are preserved in the embedding space. In practice, there is a desire to obtain low-dimensional embeddings (where the dimension is significantly

smaller than the size of the node set) for scalability reasons [GF18, cp.p.79]. Network embedding can be effectively interpreted as a dimensionality reduction technique designed to handle graph-structured data originating from non-Euclidean, high-dimensional, and discrete domains.[Cha+22, cp.p.8][SMD21, cp.]

3.2.3 Link Prediction

Over the years, numerous link prediction methods have been developed to predict connections between nodes in graphs. Methods reach from simpler heuristics to computational heavy deep learning.[WV21, cp.p.1]

Heuristic techniques are practical and straightforward methods for predicting links between nodes in a graph. They can be classified based on the number of hops they consider, including 1-hop and 2-hop neighbours or the entire graph. Hop refers to the traversal of edges between nodes in a graphs, indicating the number of steps or connections needed to reach one node from another. Local heuristics focus on the local neighbourhoods of nodes and include methods like common neighbours and Jaccard's coefficient.[Lab23, cp.]

Common neighbours is a link prediction heuristic (Equation 8) that assesses the similarity between two nodes in a graph by counting the number of common neighbours they share. For example, in a social network context, this means considering two individuals' mutual friends. The underlying assumption is that if two nodes have many common neighbours, they are more likely to be connected in the graph. Besides the simple calculation of this heuristic, only direct neighbours are considered.[Ahm+20, cp.p.4]

$$\text{Common Neighbours}(u, v) = |N(u) \cap N(v)| \tag{8}$$

Jaccard's coefficient is another link prediction heuristic (Equation 9) that considers the common neighbours of two nodes but with an additional normalization step. Instead of just counting the number of shared neighbours, Jaccard's coefficient computes the proportion of neighbours shared by the two nodes relative to the total number of unique neighbours they have. This normalization rewards nodes with relatively few interconnected neighbours, as they tend to have a higher Jaccard similarity value. The result of Jaccard's coefficient ranges from 0 to 1, with one indicating that the two nodes share all their neighbours and 0 indicating that they have no common neighbours. This measure provides a normalized similarity score that allows for fair comparisons between nodes

with different degrees of connectivity. Nodes with higher Jaccard similarity are considered more likely to be connected in the graph.[Zah16, cp.p.13f.]

$$\text{Jaccard's Coefficient}(u, v) = \frac{|N(u) \cap N(v)|}{|N(u) \cup N(v)|} \tag{9}$$

Common neighbours and the Jaccard coefficient rely on node degrees, local neighbourhood considerations, making them fast and explainable, but they have limitations in capturing complex relationships. On the other hand, global heuristics, like the Katz index and random walk methods, consider the entire network capable of capturing complex structures.[Lab23, cp.].

Katz score is a topological measure defined between any pair of nodes in a graph that captures their relationship due to the link structure. It considers the whole graph structure, including all edges between all nodes. Katz scores measure the affinity between nodes via a weighted sum of the number of paths between them. Formally, the Katz score between node i and j is defined as Equation 10.[Bon+12, cp.p.74]

$$K_{i,j} = \sum_{l=1}^{\infty} \alpha^l paths_l(i, j) \tag{10}$$

Where $paths_l(x, y)$ denote the number of paths of length l between i and j and $\alpha < 1$ is an attenuation parameter. Then computing the Katz scores for all pairs of nodes is equivalent to the following computation:

$$K = \alpha A + \alpha^2 A^2 + \dots \tag{11}$$

As seen in Equation 11, the calculation of Katz scores for all node pairs is the nearly the same calculation as the UBO calculation, except for the attenuation term alpha. For this reason, the calculation of the Katz scores is also fraught with the same problems as the calculation of the UBOs discussed in Chapter 3.1.2.[Bon+12, cp.p.74]

While heuristic methods like Jaccard's coefficient and the Katz index provide simple and intuitive approaches for link prediction, they have limitations in capturing complex relationships within large and sparse graphs. They cannot capture complex node and edge features and complex graph structures. In contrast, representation learning methods like the deep-walk or the graph autoencoder offer a more sophisticated and data-driven approach to link prediction. By learning low-dimensional embeddings for nodes in a graph, representation

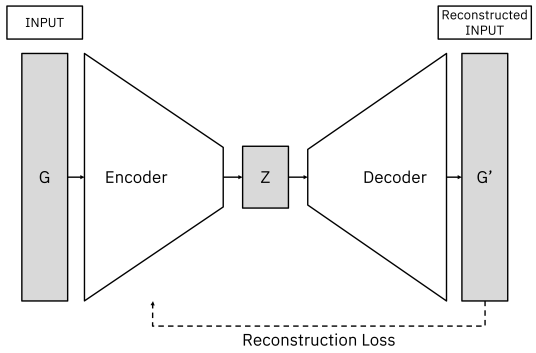

Figure 14: Autoencoder Architecture [Wen18, adapted]

learning methods can capture complicated structural patterns and generalize to unseen data more effectively. This enables them to address the challenges of more extensive and diverse datasets.[Tra18, cp.p.2][Fu+23, cp.p.3f.]

DeepWalk is a graph representation learning algorithm that aims to generate continuous node embeddings by capturing the structural information of the graph. The algorithm achieves this by simulating random walks on the graph, where nodes are sampled sequentially based on neighbourhood relationships. These random walks generate sequences of nodes, which are then used as input to train a Skip-gram model initially designed for natural language processing tasks. The Skip-gram model learns to predict the neighbour nodes given a target node in the random walk sequences. After training, the resulting embeddings map nodes to dense, continuous vectors in a high-dimensional space.[PAS14, cp.p.4-5][Wu+22, cp.p.259] For link prediction, the learned embeddings are used to measure the similarity between nodes and predict the likelihood of a link between any pair of nodes. The cosine similarity between the embeddings of two nodes is typically used to make binary link predictions, where higher similarity scores indicate a higher likelihood of a link.[Zho21, cp.p.6]

In contrast to predicting new links via the similarity of two node embeddings, new approaches, such as the Graph AutoEncoder, learn to predict new links via a complete construction of the adjacency matrix. [KW16b, cp.p.2]

An autoencoder is a specialized neural network designed to encode input data into a compressed and meaningful representation and then decode it to produce a reconstructed output that closely resembles the original input [BKG20, cp.p.2].

The typical autoencoder model architecture is constructed of three main parts, as seen in Figure 14. The encoder part of the network processes the high-dimensional input, mapping it to a lower-dimensional embedding, also called

the latent space, as shown in Equation 12. The latent space Z contains crucial information about the input in a concise and meaningful representation. Contrary to the encoder, the decoder part takes the latent variables as input and tries to reconstruct the original input $\hat{A}$ as closely as possible, as shown in Equation 13. During training, the model minimizes a variant of the reconstruction error, ensuring that the output closely resembles the input. This reconstruction objective incentivizes the autoencoder to learn meaningful and compressed representations of the input data. Since the input and the output are the same, no additional labels are required to train an autoencoder. The performance of the model relies on the calculated reconstruction error.[LRM20, cp.]

The successful concept of the autoencoder, as transferred and applied to the domain of graph machine learning by Kipf and Welling [KW16b], is inherited by the Graph Autoencoder (GAE), maintaining the encoder-decoder structure of the original autoencoder.[KW16b, cp.p.1] It uses representation learning layers like Graph Convolutional Network (GCN), Graph Sampled and Aggregated (GraphSAGE) or Graph Attention Network (GAT) to learn meaningful embeddings, including the nodes, nodes features, and edge features according to the layer possibilities. The layers are explained in detail in the following section. The decoder approximates the adjacency matrix using the inner dot product of the embeddings and calculates a link probability for every node pair based on the sigmoid function. The reconstruction loss is calculated via binary cross-entropy of positive and negative edges and their predictions.[KW16b, cp.p.1-3][Lab23, cp.][GF18, cp.p.79]

$$Z = GNN(X, A) \tag{12}$$

$$\hat{A} = \sigma(Z^T \cdot Z) \tag{13}$$

Since adjacency matrices are sparse, models are highly biased during training towards the non-existence of an edge. To prevent the reconstruction error is not calculated on the whole adjacency matrix but rather on a subset of edges. This subset includes the edges to predict (positive edges) and non-existent edges as a counterpart. By including negative samples during training, the model learns to differentiate between true and false links, allowing it to better capture the underlying patterns and dependencies in the graph. This enables the model to make more accurate and meaningful link predictions, even for sparse graphs and improves its overall performance in real-world scenarios.[KN17, cp.p.2, 6, 7]

Comparisons of the different approaches have shown that AutoEncoders perform very well when it comes to link prediction on graphs [WV21, cp.p.4-5][KW16b, cp.p.2].

3.2.4 Methods of Representation Learning

In complementing the gap in the encoder part of the graph autoencoder and finding a meaningful function that embeds all the necessary information into the latent space z, there are multiple approaches to graph neuronal network layers (GCN, GraphSAGE and GAT), each with different advantages and disadvantages.

Message passing is a fundamental concept in representation learning with Graph Neural Networks (GNNs) that allows nodes in a graph to communicate and exchange information with their neighbours. The process involves passing messages (information) along the graph's edges, where each node aggregates the received messages to update its representation. This mechanism enables GNNs to capture the graph's relational dependencies and structural information.[Gil+17, cp.p.2f.]

Passing messages is a simple four-step algorithm that can be applied multiple times to receive information from nodes further away.

Each node in the graph is assigned an initial feature vector, representing its current representation or state. At each step of the message-passing process, every node sends a message to its neighbouring nodes based on its current feature vector. A node typically sends a message that is typically a transformed version of its own feature vector, incorporating information from the node itself and its neighbours. Upon receiving messages from their neighbours, each node aggregates them to create a comprehensive view of its neighbourhood. The aggregation process is typically a summation or concatenation of the received messages, which allows the node to capture the collective information from neighbouring nodes. After aggregating the messages, each node applies an update function to its current feature vector and its aggregated message. This update function combines the information from the node's and neighbourhood's features to generate an updated node representation.[Nel23, cp.]

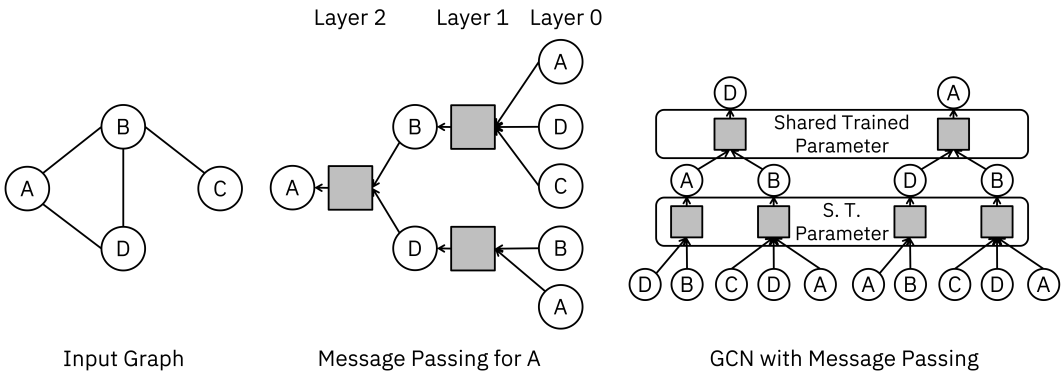

Figure 15: Message Passing and GCN [Ani23, adapted]

The message-passing process is usually performed iteratively for multiple rounds, allowing nodes to exchange information with their neighbours multiple times, as shown in the center of Figure 15. Each iteration refines the representations of nodes and helps the model capture higher-order dependencies and complex relationships in the graph. The message-passing process continues for a predefined number of iterations or until a convergence criterion is met, indicating that the node representations have stabilised. GNNs can effectively propagate information across the graph through message passing, enabling them to learn expressive and context-aware node representations.[Gil+17, cp.p.2f.][Nel23, cp.]

Based on the successful concept of message passing, further algorithms were developed that further perfect representation learning by borrowing and integrating successful concepts from other areas of ML.

GCNs are a type of neural network designed to operate on graph-structured data. The key idea behind GCNs is to leverage the local neighbourhood information of each node. Each node's features are updated by aggregating and combining features from its neighbours. The convolutional operation in GCNs can be thought of as a message-passing process, where nodes send messages (feature information) to their neighbours, and each node updates its features based on the received messages. A critical difference between GNNs and GCNs is sharing weights/parameters through the different layers, as shown on the right side of Figure 15. GNNs share their learned parameters through the layers, whereby the layers of GCNs have their own learned parameter, seen via the weight matrix W^l, similar to the convolutions of an CNN known from image processing.[KW16a, cp.]

Mathematically, the GCN layer can be expressed with the following Equation 14. [KW16a, cp.][Ani23, cp.]

$$H^{(l+1)} = \sigma\left(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}H^{(l)}W^{(l)}\right) \tag{14}$$

Where:

- $H^{(l+1)}$ represents the output feature matrix of the GCN layer $l+1$, which contains the updated node representations.
- $H^{(l)}$ is the feature matrix of the $l-th$ layer, representing the node embeddings.
- $W^{(l)}$ is the weight matrix of the GCN layer l, which is used to learn the transformations between the input and output feature spaces.
- $\tilde{A}$ is the graph's adjacency matrix with added self-loops to incorporate self-information
- $\tilde{D}$ is the degree matrix of $\tilde{A}$, which is a diagonal matrix with the degree of each node along the diagonal.
- σ is a non-linear activation function, typically Rectified Linear Unit (ReLU)

The GCN layer aggregates information from neighbouring nodes by multiplying the feature matrix $H^{(l)}$ with $D^{-1/2}\tilde{A}D^{-1/2}$, which is a normalisation step. The weight matrix $W^{(l)}$ learns to combine the aggregated features to produce the updated embeddings $H^{(l+1)}$. This process is iterated over multiple layers, allowing GCNs to capture and propagate information from distant nodes in the graph. In addition, the GCN approach is purely transductive by design, as the embedding is generated from the frequencies of all values during training and not derived during inference.[KW16a, cp.][Ani23, cp.]

GraphSAGE is a graph-based machine learning algorithm used for inductive node representation learning in graph-structured data.

Key components of GraphSAGE are sampling, aggregation and applying pooling functions. First, GraphSAGE employs a neighbourhood sampling strategy to handle large graphs efficiently and thereby enables the inductivity. It samples a fixed-size neighbourhood for each node during training. This sampling is done in a manner that ensures a good representation of the local graph structure. After sampling the neighbourhood for each node, GraphSAGE aggregates the feature information from the neighbours to create a comprehensive representation for each node. The aggregation step combines information from the node's features and features from its sampled neighbours. GraphSAGE uses different pooling

functions (e.g., mean pooling, max pooling) to aggregate the features from the sampled neighbours. The pooling function determines how the features are combined to form the final representation of each node. Unlike the previously presented GNNs, GraphSAGE does not rely on learning parameters based on matrix factorisations, and there while learning the best aggregation, it can generalise its learned knowledge to new unseen graphs.[HYL18, cp.][Ani23, cp.]

GATs are a type of GNN that leverage attention mechanisms to perform adaptive and context-aware neighbourhood aggregation. GAT was introduced to address the limitations of fixed-weight neighbourhood aggregation in traditional GNNs like GCNs and GraphSAGE. In GAT, the key idea is to allow each node to learn its attention coefficients for aggregating information from its neighbours. These attention coefficients are calculated using an attention mechanism inspired by the concept of self-attention in natural language processing. The attention mechanism enables nodes to assign different levels of importance to each neighbour based on their relevance to the node's representation learning task.[Vel+17, cp.][BAY21, cp.p.1]

Like the other GNNs, the GAT starts with receiving an initial input embedding, typically represented as a high-dimensional vector. GAT calculates attention coefficients for each neighbour of a node. These coefficients represent how much attention or importance the node should give to each neighbour during aggregation. The attention coefficients are computed based on the similarity between the node's embedding and the embeddings of its neighbours. Using the attention coefficients, GAT performs a weighted aggregation of the embeddings of neighbouring nodes. Instead of using fixed weights, like with GCN, GraphSAGE, each neighbour's embedding is multiplied by its corresponding attention coefficient, and the resulting weighted embeddings are summed to obtain the aggregated representation for the node. A self-loop is added to consider the node's information during aggregation to ensure that the node's embedding is included in the aggregation process. GATs often employs multiple attention heads to capture different aspects of the node's neighbourhood. The outputs of multiple attention heads are concatenated or averaged to produce the final aggregated representation for the node. After aggregation, a non-linear activation function (e.g., ReLU) is applied to introduce non-linearity into the node's representation.[Vel+17, cp.p.3-5]

The GAT can be applied in both transductive and inductive settings due to its shared attention mechanism, which enables it to operate on graphs without requiring upfront access to the global graph structure or features of all nodes

during training. In addition, GAT can also include edge attributes in the calculation, which is impossible with the previously mentioned models.[BAY21, cp.p.2-4][Ani23, cp.]

	Transductive	Inductive	Node Features	Edge Features
GCN	✓	✗	✓	✗
GraphSAGE	✓	✓	✓	✗
GAT	✓	✓	✓	✓

Table 1: Graph Neural Network Comparison

Finally, Table 1 compares relevant attributes of the different layer architectures required for link prediction. In particular, whether they are suitable for transductive or inductive problems and what features they include in their calculation are considered, which will be relevant for the design of the artefact.

4 Artefact Development

Following an in-depth exploration of the prevailing problem and the business interest associated with the identification of UBOs, as well as a comprehensive review of the state-of-the-art in graph machine learning techniques, the research proceeds to adopt the DSR process. In line with this approach, an artefact is developed to address the existing challenge using graph machine learning. The artefact development is structured into distinct phases, as Figure 16 illustrates.

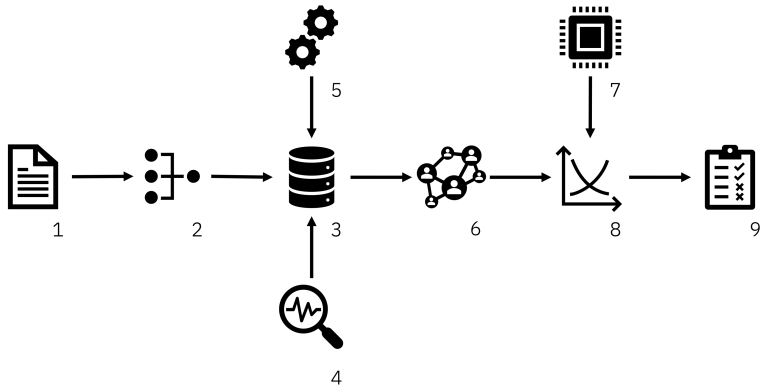

Figure 16: Steps of Artefact Development

These phases are:

- Phase 1: Acquisition of a suitable dataset that fulfills qualitative, quantitative, and domain-specific requirements.
- Phase 2: Preprocessing and transformation of the dataset into a graph.
- Phase 3: Storage of the graph in a graph database for efficient storage and management.
- Phase 4: Preliminary exploratory data analysis, including an assessment of dataset characteristics and ownership structures.
- Phase 5: Augmentation of the dataset using various methods to enhance size, realism, and essential information.
- Phase 6: Generation of a dataset conforming to the necessary format for machine learning models, leveraging prepared ownership structures.
- Phase 7: Definition of suitable machine learning models using techniques from Chapter 3.2.3 to address the problem.

- Phase 8: Model training and hyperparameter tuning, leading to the creation of the model as artefact.
- Phase 9: Comprehensive evaluation, including demonstrations and problem-specific assessments using predefined test scenarios.

The following technologies are used to conduct the case study. Python 3.9 is the general programming language for data preprocessing, data augmentation and model handling. Neo4j, a graph database, is used for efficient management, storage and retrieval of ownership graphs. The machine learning model is defined and trained using the Python libraries PyTorch and PyTorch Geometric, which specializes in graph machine learning.

4.1 Dataset

Finding and using a suitable dataset is crucial for every machine-learning task since every model can only learn from the data. In order to adequately answer the research question and to train an ML model as best as possible to identify UBO, it needs a dataset that fits the problem in terms of quality, quantity and content.

Data quality is one of the most critical points, saving much work in preprocessing and ensuring well performing models. The most important factors in data quality, based on the Dataset Definition Standard (DDS), are the representativeness of the data, its consistency and reliability. Data representativeness is the extent to which the data can accurately represent the operational domain of the given problem. If, for example, only a specific group of people is interviewed in a survey, the dataset will only represent the surveyed group, and it may not be possible to make assumptions and results for the entirety. Data consistency is a critical aspect of any dataset used for machine learning. Consistency ensures that the data is accurate and coherent, making it possible for the machine learning model to generate meaningful insights and make accurate predictions. Data reliability hinges on the credibility and relevance of the data within the operational domain. It is pivotal in seeding confidence during model testing and performance assessment. Data reliability is imperative as it directly influences the trustworthiness of the machine learning model's predictions and outcomes. This reliability is, in turn, safeguarded by the credibility and appropriateness of the data sources utilized.[Cap+21, cp.p.4-10]

In machine learning, dataset size plays a crucial role in the performance and generalisation of models. Generally, a more extensive dataset provides more

diverse and representative samples, enabling the model to capture complex patterns and relationships within the data. As the dataset grows, the model becomes more exposed to various scenarios and can better handle unseen instances during testing, enhancing its ability to make accurate predictions on new data during inference. A more extensive dataset offers a richer source of information, reducing the risk of overfitting and ensuring improved model generalisation. In general, the accuracy of a trained model is logarithmically related to the size of the dataset. Smaller datasets quickly get an improvement in accuracy as the dataset grows, but this effect weakens when the dataset size reaches a certain point. However, it is essential to strike a balance, as massive datasets can also lead to higher computational costs and training time. Thus, carefully selecting an adequately sized dataset is vital to achieving a well-performing machine learning model.[SMF20, cp.p.15f.][Tab+19, cp.p.5]

In addition, the general problem definition and the literature review result in several content-related requirements for the dataset. The dataset should represent different ownership structures, including companies and individuals. Ideally, these should already be available as a graph or can be converted into a graph by preprocessing. Another critical point is the presence of essential features. These include, in particular, the ownership characteristics determined in Chapter 3.1.1, which are necessary to calculate beneficial owners and ultimate beneficial owners. In addition, the data should be centrally accessible to avoid having to assemble or scrape it from several sources in order to minimise this work's implementation effort. Finally, the data should be freely available and licensed for scientific use.

As described in Chapter 3.1, companies in many parts of the world are legally required to disclose their beneficial owners to the government. For the purpose of transparency, in many cases, the collected data is made available in excerpts or entirety. However, according to the WorldWideWeb Foundation's report on open data, 72% of countries make their data available free of charge. However, only 32% of all data is machine-readable, and only 17% is licensed for open use [Ope17, cp.p.29].

The German Handelsregister clearly illustrates this condition. Every company registered in Germany, and information about it can be found via the German Handelsregister. However, data on each company must be retrieved individually; the requests are limited to 60 per IP address per hour. Moreover, the retrieved data is usually in a .pdf document, which is not machine-readable and relevant information would have to be extracted at great expense for an analytic process.[Han23, cp.]

These complex problems have resulted in several initiatives that compile current data, sometimes from different data sources, and combine them into one dataset. These initiatives are primarily companies such as Dijk [Dij23] or NRG-Metrics [NRG22], which sell the compiled datasets to their customers, as the need for such data is also constantly increasing in the industry. However, access to such aggregated datasets is highly restrictive.[Lor19, cp.p.58]

The OpenOwnership (OO) initiative advocates for beneficial ownership transparency as a critical element in facilitating a well-functioning economy and society. By promoting public registers of beneficial owners, OO seeks to provide high-quality data about company ownership, control, and profit beneficiaries, aiming to combat corruption, minimize investment risk, and enhance global governance. OO collaborates with numerous countries, assisting in beneficial ownership reforms and supporting the establishment of registers.[Ope23, cp.]

One of the projects of the OO is the OpenOwnershipRegister, which contains structured data on ownership structures and their beneficial owners scraped from four European country registers, the UK, Denmark, Slovakia and Ukraine. This dataset can be downloaded in bulk and contains approximately 23 million statements. These statements are defined according to the Beneficial Ownership Data Standard (BODS) and thus ensure a high uniformity and data quality of the dataset. The dataset contains all essential elements of an ownership structure, including companies (entities), individuals, and their ownership relationships defined by capital shares, voting rights and other values. Furthermore, the dataset is licensed under the Open Data Commons Attribution License and can be used freely.[Own23, cp.]

As a result, this dataset meets the previously defined requirements for quality, quantity and content and is used for the following analyses and developments.

4.2 Data Preprocessing

Data preprocessing is a fundamental and crucial step in data analysis and machine learning. It involves cleaning, transforming, and organising raw data into a structured format suitable for further analysis or modelling. Proper data preprocessing ensures that irrelevant or noisy information is removed, missing values are imputed, and data is appropriately scaled or normalised. By preparing the data in a consistent and standardised way, potential biases and inconsistencies are minimised, enabling more accurate and meaningful insights to be derived from the data. Data preprocessing lays the foundation

for successful data analysis and developing suitable machine learning models, enhancing overall performance and facilitating informed decision-making in various domains.[Aga15, cp.p.30f.]

The primary goal of data preprocessing for the selected dataset is to transform text-based data into a usable graph structure. In addition, the corresponding features are prepared for later processing, and the dataset is cleared of duplicates.

The dataset can be downloaded in its entirety from OpenOwnership. The file is about 21GB and is defined in the JSON Line standard [War23, cp.]. Each file line is a self-contained json structure that defines an entity, a person or an ownership relationship and is defined according to the BODS. There are about 23 million of these statements in the raw dataset.

Since the individual statements are not sorted by category, they are first divided into the aforementioned groups (entity, person, ownership and relationship). The first selection of relevant features and the first feature engineering is done within these groups.

Each statement, regardless of whether it is an entity, person or ownership relationship, has a statement id as a unique identifier, which is adopted and will continue to be used as an identifier later in the data augmentation and the creation of training and test datasets.

For entities, the name and the founding date are selected in addition to the statement ID. Entities with a closing date, i.e. have already been closed and no longer exist, are deleted from the dataset and excluded from later consideration. For persons, the name, the birth date and the country of origin are extracted in addition to the statement ID.

For the ownership statements, there is also the statement ID. In addition, there is the specification of „subject“and „interested party“, each containing a statement ID and thus defining the edge between two nodes. „Subject“denotes the directed edge's target node, an entity. „InterestedParty“is the beneficial owner of the subject and can be of type entity or a person. Furthermore, this definition defines the „interests“, i.e. the exact expression of the ownership relationship between the two nodes. The interests are stored in a list and contain all types of ownership the source node has at the target entity. The BODS defines 21 interests that an owner can exercise on an entity. These include, for example, capital shares, voting rights, but also various legal and organisational interests.

The following five ways of exercising ownership are excerpted from the 21 and illustrate this diversity.

1. Appointment of Board: This interest grants a person the absolute right to appoint members of the board of directors, giving them significant influence and control over the company's strategic decisions.
2. Senior Managing Official: This interest is gained through employment and provides a controlling interest in the entity, allowing the person to exercise control over the company's management.
3. Trustee: A trustee is responsible for administering a trust for the benefit of a third party. They hold the legal title of the trust property and have a fiduciary duty to act in the best interest of the beneficiaries.
4. Rights to Profit or Income: This interest grants economic rights beyond typical ownership structures. It allows the individual to receive profits or income from the company's activities beyond their ownership stake.
5. Control by Legal Framework: This interest is established through a legal framework, such as legislation, to guide entities linked to the state. It grants control over an entity based on specific laws and regulations.

These various types of interests in a company play a crucial role in shaping the governance and decision-making processes within the organisation.

The variables, capital shares, voting rights and other powers define the bare minimum to calculate the ultimate beneficial ownership of an ownership structure, as already defined in Chapter 3.1.1. In order to provide these in a machine-usable format, the provided list of interests is converted into a sequential enumeration variable.

In many cases, the ownership information in the dataset is provided in the form of intervals, indicating an owner's capital shares and voting rights. These intervals are expressed as 25%-50%, 50%-75%, or 75%-100%, with open or closed limits specified. Notably, the interval for 0%-25% is absent since the dataset exclusively contains beneficial owners who, by definition, hold more than 25% ownership.

To convert these interval representations, the average of the interval limits is calculated and stored, considering whether the limits are open or closed. Occasionally, when no interval is given, an exact proportion may be specified, which can be directly utilized. In cases where no proportion is provided, 0 is assigned. By specifying the interests in intervals in the raw dataset and calculating the averages, all shares in a company may exceed 100. However,

since this does not change the actual ownership structure, it does not pose a problem in the further course.

However, converting other interests is more complex, as the list may contain an arbitrary number of other interests without specific values for ranking. To address this, the number of other interests is counted and recorded as a numerical feature. This approach allows for quantifying the power exercised through ownership over other interests. Consequently, beneficial owners with more other powers are shown to exert more significant influence through their ownership positions.

The sorted and converted datasets are inserted into a Neo4j database for efficient storage, modification and retrieval. Beginning with the nodes and later adding the existing ownership edges to complete the ownership structures of the dataset as a graph.

Graph databases, exemplified by Neo4j, play a crucial role in data science, especially when dealing with intricate and interconnected datasets. Distinguished from traditional relational databases, graph databases excel in capturing and representing relationships between data points, rendering them highly suitable for applications involving network structures and social interactions [Bes+19, cp.p.1-3][TRM21, cp.p.1-3]. Neo4j, as a graph database, offers several advantages that prove beneficial for graph data projects. Its schema-less nature allows dynamic data representation with varying structures, imparting flexibility and adaptability to evolving datasets. The query language, Cypher, tailored for graph-based queries, makes traversing relationships and extracting relevant information intuitive and efficient. Moreover, Neo4j performs well in handling large-scale networks and complex graph operations, ensuring rapid querying and analytics, thereby facilitating real-time decision-making.[RWE15, cp.]

Following the complete data insertion, an active check for duplicates is performed, despite OpenOwnership's claim of eliminating duplicates from their dataset. It has been observed that some duplicates still exist, particularly among persons. To identify duplicates, nodes' properties are analyzed. Specifically, persons are deemed duplicates if their name, date of birth, and country of origin are identical. Similarly, entities are recognized as duplicates based on their name and foundation date. Unlike conventional methods that involve deleting duplicate entries, duplicates are merged in this case [RD08, cp.p.768]. For instance, if two identical persons are found in different ownership structures, they can be consolidated into one without impacting the underlying ownership structures. This merging approach ensures a more comprehensive

and accurate representation of the data while preserving the integrity of the graph structures.

4.3 Exploratory Data Analysis

Exploratory Data Analysis (EDA) is an essential step in any data-driven research project, enabling a comprehensive understanding of the dataset's characteristics, patterns, and underlying relationships. Its primary purpose is to inform subsequent analyses, model selection, and data preprocessing techniques by identifying data quality issues, outliers, and potential biases.[Kom+16, cp.p.185]

This chapter performs a comprehensive EDA journey, beginning with a broader perspective on the graph data and gradually focusing on ownership structures.The goal of this EDA is to build an understanding of the graph resulting from the data preprocessing and to find options to enhance the graph further for the later machine learning model training.

4.3.1 Graph Analysis

The preprocessed ownership graph consists of 15.5 million nodes. Within this network, 8.3 million nodes are categorized as entities, while the remaining 7.1 million nodes represent individuals classified as persons.

The relationships between these nodes are defined by 9.1 million edges, each indicating beneficial ownership. Remarkably, most of these edges (8.5 million) connect a person to an entity, showing the direct ownership of entities by people. In contrast, there are only 550.000 edges connecting two entities, denoting instances where one entity possesses beneficial ownership of another entity.

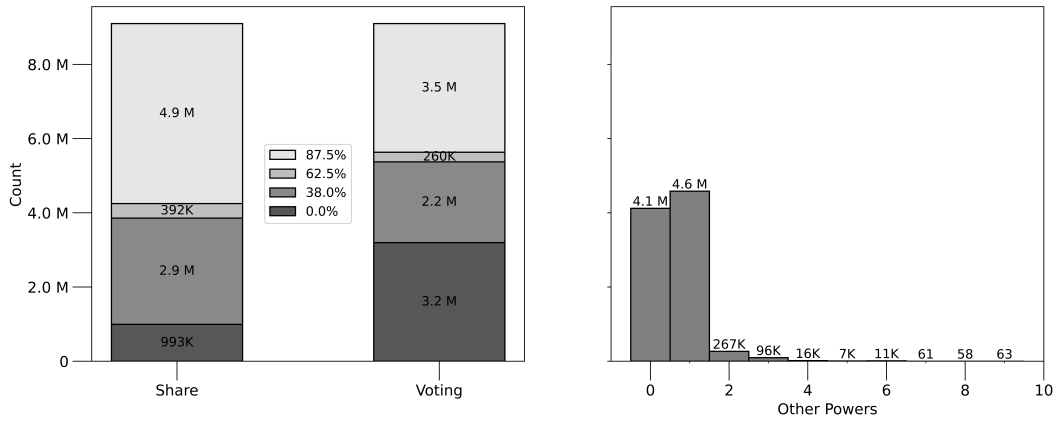

Figure 17: Edge Attributes Distribution

The properties of the edges in the graph further reveal interesting insights, displayed in Figure 17, please keep in mind that the y axis is scaled logarithmically to increase visibility of smaller values. The dataset has capital shares and voting rights, with almost all values concentrated around 37.5%, 62.5%, and 87.5%. Notably, there is a gap at 12.5%, which would be the average of the interval between 0% and 25%. This discrepancy arises due to the exclusion of non-beneficial owners from the dataset. Consequently, instances with zero shares are present when individuals hold no voting rights or capital shares. The chart also shows that the option of specifying shares/voting rights directly via the BODS is hardly used.

Regarding the „other power“property, many entities exhibit no other forms of power. A sizable proportion possess minimal power, with only one or two additional forms of influence. As the spectrum progresses, the number of entities with higher levels of power outside of the norm decreases rapidly, highlighting a significant imbalance in this particular situation.

In the second step of considering the edge attributes, the dependencies of the attributes among each other are now reviewed, for which the Pearson correlation is used. The Pearson correlation describes the strength of the linear relationship between two variables. The correlation is given as a value between -1 and 1, where 1 stands for a robust linear relationship between the variables and -1 for a strong negative relationship. A correlation of 0 indicates that the variables are not related.[ER10, cp.p.156]

Looking at the correlations of the edge attributes (see Appendix Table 4), a weak correlation can be seen between the capital shares and the voting rights and also between the voting rights and the other powers. Both correlations are

around 0.4, indicating a weak linear relationship. Between the capital shares and the other powers, however, there is a correlation of -0.01 and thus no dependency of the variables.

Finally, the statistical significance of the correlations is evaluated using a t-test since the present values are only a sample of the real world.[SBH17, cp.p.384] For this purpose, a significance level of 5% is set and compared against the calculated p values [POD08, cp.p.532]. The p values of all calculated correlations are 0, which speaks for the statistical significance of the statements made. The exact p-value of 0 can be explained by the very large sample size of 9.1 million.

The general one-share-one-vote assumption [PPS21, cp.p.3][BL08, cp.], which assumes that a capital share is simultaneously associated with a voting right, cannot be assumed due to the lack of correlations, which requires a separate consideration of the control options.

4.3.2 Ownership Structure Analysis

After a preliminary examination of the overall graph characteristics, the investigation turns towards a more focused analysis of the underlying ownership structures that emerge from the graph data. A count of approximately 7 million ownership structures can be found within this dataset. Among these, roughly 5.5 million ownership structures are trivial, whereas the remaining 1.5 million manifests themselves as complex ownership structures. Trivial ownership structures contain only one natural person, which is, per definition, the one and only UBO, disregarding the complexity of the whole network. The trivial structures are intentionally excluded from the following analytical process due to their inherent clarity, preventing the requirement for explicit UBO calculations.

The dataset introduces different ownership structures with diverse levels of depth, signifying the extent of relationships and ownership hierarchies through companies. The depth of an ownership structure ranges from a minimal level of 1 to a depth of 19, as seen in Figure 18. Whereby a depth of 1 represents direct ownership of a company by Persons, and a depth of 19 describes scenarios where a target entity is owned through a multi-layered chain of up to 19 distinct entities and persons. An interesting aspect of this examination is the evident contrast among the dataset's depth distribution. A closer look reveals that roughly one million ownership structures exist at depth 1, indicating the prevalence of straightforward, directly-owned companies. In contrast, the upper limit of depth, such as depth 19, is notably low, with only two ownership structures recorded in the dataset. The structure decline as depth increases follows an

exponential trend, which becomes apparent when reviewing logarithmic scaling on the Y-axis for better representation. The rationale behind this trend is easy to understand: constructing a corporate network with ownership links at lower levels is generally more straightforward than weaving a complex ownership network that extends to higher depths.

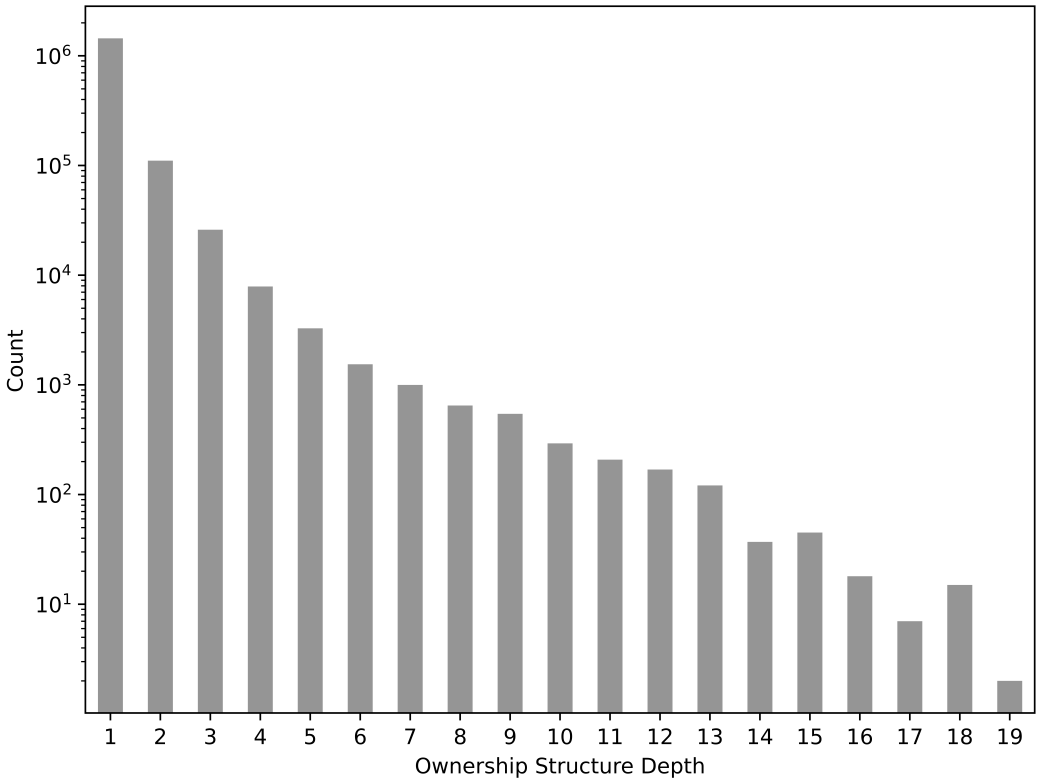

Figure 18: Number of non-trivial Ownership Structures by Depth

It is also essential to consider not only the depth but also the number of nodes in an ownership structure. For this purpose, boxplots of the number of nodes per depth are shown in Figure 19. The general increase in the number of nodes with increasing depth of the ownership structures is visible. However, it does not represent the exponential increase in nodes with increasing depth expected from the theoretical Equation 2. However, this deviation can also be explained reasonably, as the present ownership structures only contain beneficial owners, don't include all owners, and therefore do not meet the targeted complexity. It is also apparent that there are considerably more outliers in the shallow depth range than in the deeper structures. It is also straightforward to see that the interquartile range is often small, which indicates a central tendency of the data. The average and the interquartile range, which encapsulates the middle 50% of the data, are situated in close proximity. This proximity indicates a concentrated distribution in which most data points cluster around the central

value. Additionally, the limited presence of outliers suggests that relatively few data points deviate significantly from the overall trend. This pattern indicates a data distribution characterized by a consistent and compact arrangement with minimal variability and extreme values.

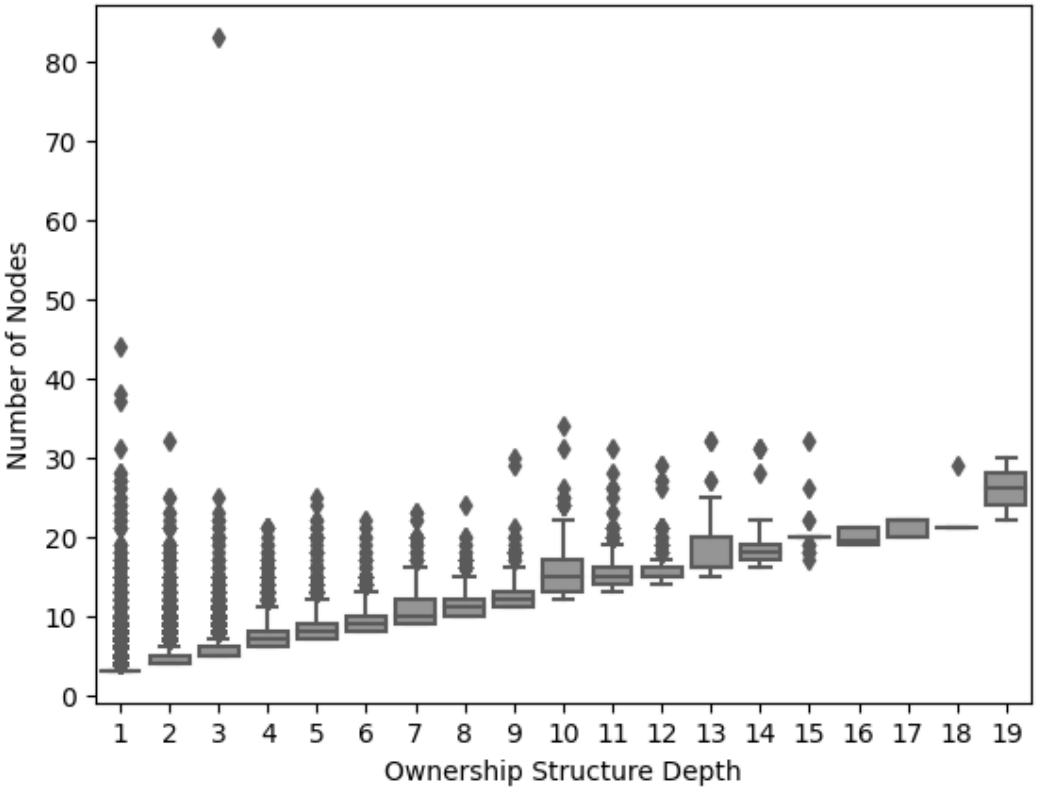

Figure 19: Number Nodes by Depth

The investigation of depth and node count within ownership structures is foundational yet needs to be more comprehensive in capturing the complexity of such structures. For this reason, the Average Incoming Node Degree (AIND) of the entities in an ownership structure is used to evaluate the complexity. This metric unveils the extent of inbound ownership relationships each entity possesses, thereby discerning the density of the graph, be it sparse or dense. A critical distinction must be made between persons and entities since persons cannot have any incoming ownership relationships, which means they are excluded from this calculation. For example, an average incoming node degree value of 1 means that each entity has exactly one incoming edge, and the ownership structure is a chain. As shown in Figure 20 an average value of 2, for example, means that each entity is, on average, owned by two other entities or people. A higher value represents more connections between nodes in the graph and, thus, more complex ownership graphs.

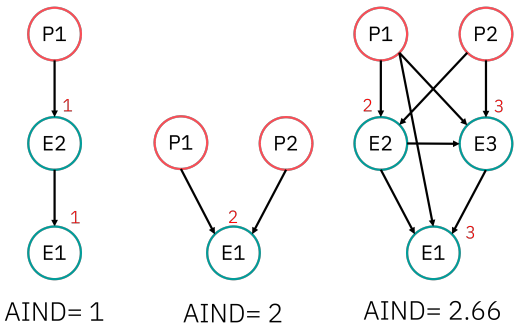

Figure 20: Examples of AIND

Across all non-trivial ownership structures in this dataset, the average is around 2, with a standard deviation of 0.46. The 25% and the 75% quartiles are also about 2, suggesting a central and dense distribution with few outliers outside of that. The minimum average is 1.05, and the maximum is 43. These values indicate a certain fundamental complexity of the available structures, which are generally well-suited as a data basis for analyzing and solving the research question posed.

Finally, a search for cyclic ownership structures, as described in Chapter 3.1.2 is performed. There are 1300 cyclic substructures in the 1.5 million non-trivial ownership structures, including smaller cross-ownership structures and larger circle structures up to a length of five. The fact that cyclic ownership structures are present in the used dataset shows the consistency with reality and the need to address problems such as calculating UBOs on cyclic structures with new procedures.

4.4 Data Augmentation

Data augmentation is a crucial technique in modern machine learning to enhance the performance of models by increasing the quantity, quality, and diversity of training data. Gathering large-scale, diverse and representative data can be time-consuming and resource-intensive. More training data can lead to better generalization of machine learning models. Data augmentation addresses this limitation by generating additional training samples through various techniques. Data augmentation is mainly well-known in image processing, where images are rotated, distorted, cropped or discoloured to increase the size and diversity of the datasets.[SK19, cp.p.7ff.] In imbalanced datasets, the number of instances in one class significantly outweighs the number of instances in another class,

leading to biased model training and potentially poor performance in the minority class. By applying data augmentation to the minority class, we can artificially increase the number of instances and balance the class distribution, thereby improving the model's ability to learn from both classes equally [Lee+18, cp.p.5-6]. Augmenting data allows for better model generalization, even with smaller and poorly representative datasets, and reduces the need for complex model architectures, making data augmentation an essential tool in machine learning tasks.[MM22, cp.p.1f.]

Like data augmentation on images, texts or tabular data, data augmentation can also be applied to graph data. Structure-oriented graph data augmentation deals with augmenting the structure of graphs. This includes, the addition of nodes, edges or whole subgraphs. For attribution augmentation, node and edge attributes can be augmented. For example, we can apply random noise or alter the values of features to create variations of the original graph. Both structure-oriented and feature-oriented graph data augmentation techniques can be combined to enrich the training data and improve the performance of graph-based machine learning models.[Din+22, cp.p.4-5]

This case study uses graph data augmentation to enrich the generated graph to adjust it to the presented problem better. Especially the skewness and imbalance of usable ownership structures will be approached. Moreover, additional non-beneficial owners are added to the graph to bring the given graph data closer to the real world problem of complex ownership structures. Finally, the graph is enriched with the calculation and addition of UBO edges, which are used as the ground truth of the machine learning model.

4.4.1 Subgraph Augmentation

From the preceding exploratory data analysis, the dataset of non-trivial ownership structures is significantly left-skewed. Many more ownership structures have a lower depth (e.g., 1 or 2) than a higher depth. Since machine learning models might be biased towards a more distinct data structure when trained using an accordingly skewed dataset, this issue needs to be addressed prior to training [Meh+19, cp.p.2][MBM22, cp.p.4].

In the context of the ownership dataset, a bias towards flatter structures might, for example, result in the ML model not learning to determine the actual UBO, but rather tending towards individuals close to the target entity. Additional ownership structures are augmented from existing ones to balance the dataset.

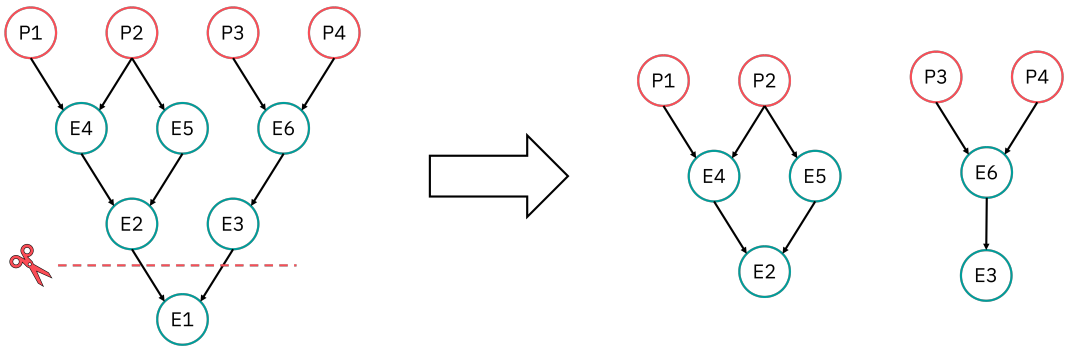

Figure 21: Subgraph Augmentation Example

For this purpose, the existing structures in the dataset are analyzed based on their complexity. To achieve this goal, the AIND of the entities of an ownership graph is used. Figure 21 shows the subgraph augmentation procedure very clearly. The starting point is a non-trivial ownership structure with many branches between the individual entities, shown on the left. In the truncation process, the structure is cut between level 0 and level 1 and the remaining connected paths are copied. This creates two new non-trivial subgraphs that further enrich the dataset. To prevent the generation of new subgraphs, which are considered trivial, only ownership structures that exceed a certain threshold of the AIND of 1.5 are considered valuable graphs for augmentation. Suppose the corresponding graphs with these properties are found. In that case, the initial structure, including all node and edge features, is copied and truncated at certain levels, which generates new ownership graphs. By truncating the original graph at different levels, it is possible to augment multiple new ownership graphs from one single original graph.

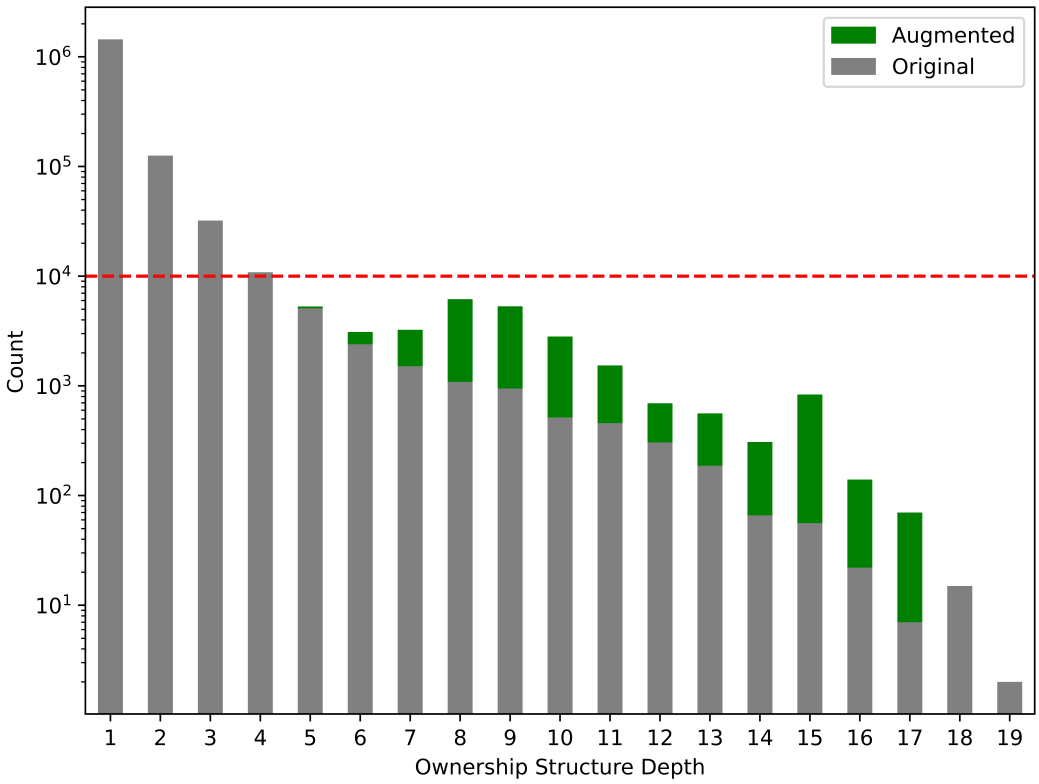

Figure 22: Number of non-trivial Ownership Structures by Depth after Augmentation

The subgraph augmentation process in the context of non-trivial ownership structures leads to observable improvements in the data distribution, as evident from Figure 22. Notably, the augmentation of new structures (indicated in green) addresses the skewness of the dataset, particularly for ownership structures with medium to high depths. Furthermore, implementing a limit (indicated in red) on frequently occurring structures, restricting them to 10.000 occurrences, contributes to a more balanced dataset.

4.4.2 Adding non Beneficial Owner

Following the presented dataset, the dataset utilized for this case study encompasses information on companies and their beneficial owners. In order to replicate the complex ownership structures found in real-life situations, a process of ownership augmentation is conducted to introduce non-beneficial owners into the graph. To achieve this, the existing owners of each company are analyzed, taking into account their respective voting rights, capital shares, and other powers. These values are summed up to determine the missing values filling up the 100% ownership. Subsequently, a variable number of new

owners is generated for each company, with the count drawn from a normal distribution with a mean μ of 5 and a standard deviation σ of 5. In addition, it is ensured that the number of new owners is always greater than 0. The missing ownership shares are then distributed among the newly generated owners randomly. Meanwhile, a normal distribution is employed to model the distribution of other powers, which are primarily small based on previous observations. In order to preserve the integrity of the ownership structure, no new owner must be granted a share exceeding 25% in voting rights or shares. This condition ensures that such owners do not attain beneficial ownership status. The defined new owners are inserted into the Neo4j database for each respective company after generating the relevant ownership values. This process effectively augments the ownership graph, yielding a more comprehensive and realistic representation of ownership structures for further analysis.

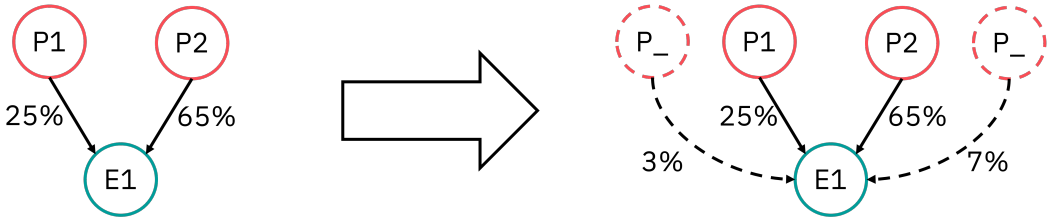

Figure 23: Ownership Augmentation Example

Figure 23 shows the procedure for owner augmentation. Starting from an ownership graph with one target entity and two beneficial owners, it is easy to see that the shares of the two beneficial owners reach less than 100%. For this reason, two new non-beneficial owners are added to the ownership structure to complement the shares.

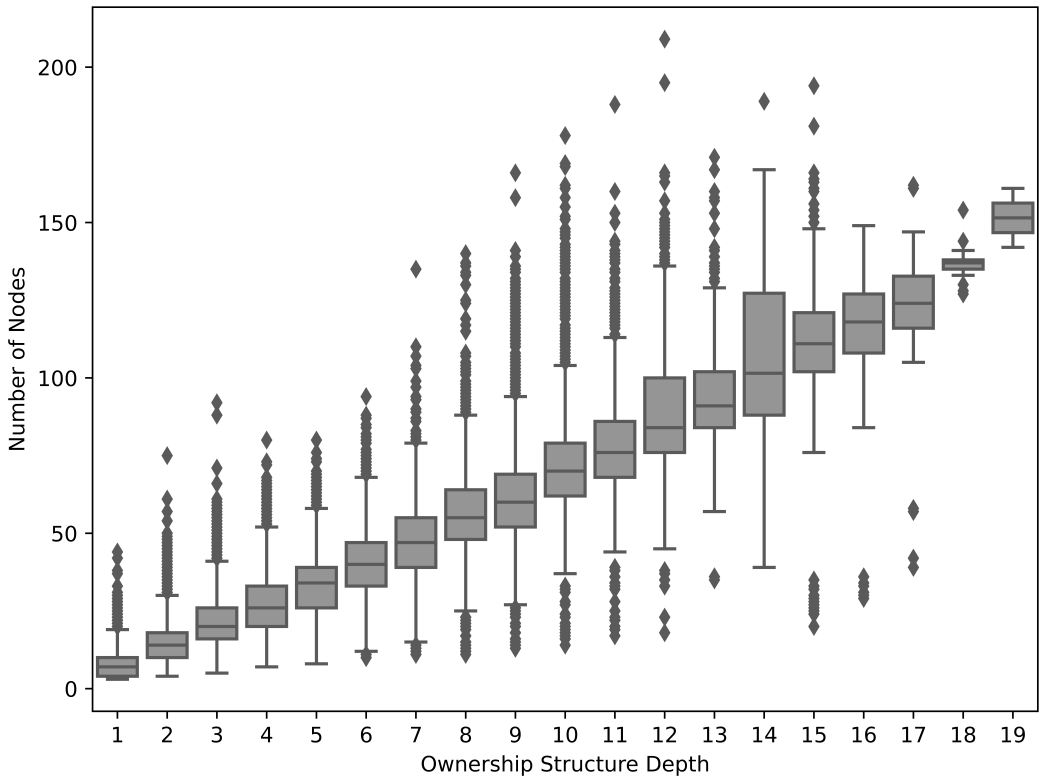

Figure 24: Number Nodes by Depth after Augmentation

The augmentation of non-beneficial owners demonstrates a good enhancement in the complexity of the dataset. A comparison of the mean node count per ownership structure depth (refer to Figure 24) against the initial situation (refer to Figure 19) shows that the mean node count at every depth has experienced an increase. The interquartile range increased through augmentation, yielding a broader distribution of data points. Moreover, more outliers above and below the median further ensure heightened diversity within ownership structure groups. This augmentation additionally influences the AIND. This metric has increased from 2 to around 6.8, which means that each entity now has, on average, more incoming ownership edges. In addition, the standard deviation of this value has increased from 0.46 to around 3.78, which again articulates an increased diversity within the ownership structures. The new non-beneficial owners add the desired complexity and variety to the existing ownership structures to mimic real-world complexity and real problem scenarios.

4.4.3 Ultimate Beneficial Owner Calculation

As a third step of the augmentation, the UBOs edges of the individual ownership structures of the dataset are calculated since these are also not included in the original dataset.

The UBOs of an ownership structure are calculated using the algorithmic approach described in Chapter 3.1.1. For this purpose, the ownership structures are retrieved from Neo4j and transferred into three weighted adjacency matrices. Each of the three matrices represents the exercise of power through capital shares, voting rights or other forces.

As described, the ascending matrix powers are summed until the result converges. In descending order of importance, they are evaluated. If more than one owner can be considered as UBO for the capital shares, the voting rights results are evaluated and then the results of the other powers. Thus, the UBOs of each ownership structure are identified and entered as the corresponding edge in the Neo4j.

If the matrices fail to converge during the calculation, the calculation of the next lower level is used. If all three matrices fail to converge, calculating the UBOs becomes impossible, and the ownership structure is sorted out.

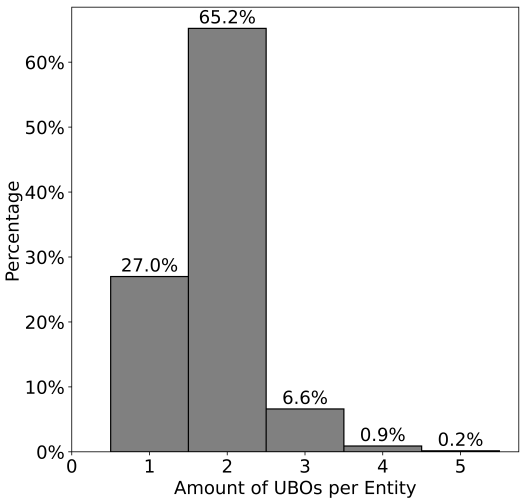

Figure 25: Amount of UBOs per Entity

Among the database's 1.6 million non-trivial ownership structures, 2.9 million UBOs were identified. This observation might initially seem paradoxical, given that each ownership structure should inherently possess only a singular UBO. Nevertheless, an exception to this norm arises when n individuals collectively

exercise equivalent ownership across all three categories on the target entity. In such instances, n distinct UBOs can arise.

Reviewing the distribution of UBOs per ownership structure in Figure 25 it can be seen that nearly two-thirds of all structures have precisely two UBOs, approximately 27% are characterized by a single UBO, and approximately 7.7% demonstrate the real-life complexity of this problem by having more than two UBOs. Ownership structures where the calculation of UBOs does not converge were not found.

4.5 Machine Learning Approach

After preparation, investigation, and augmentation of the dataset, the following section describes the design and implementation of a machine learning model in terms of the DSR artefact.

The model should be designed to predict the UBO relationship of a person to a target entity as a link between the two nodes. Due to the variety of node types (persons and entities), the model gets a heterogeneous graph as input. Special attention has to be paid to this during design and implementation. In addition, the described problem is an inductive graph machine learning problem since no fixed graph is available at the time of training, and it is still evolving with each new company registration over time. This results in the requirement that the model must also be able to make valid statements for previously unseen nodes.

These requirements lead to the following machine learning problem, which will be tackled by developing an artefact in the following section.

Inductive link predictions on heterogeneous graphs

4.5.1 Dataset Preparation

The prepared and augmented dataset of the different ownership structures continues to be stored in the Neo4j graph database. Final steps must be taken to make them available for machine learning model training.

In machine learning, a dataset is typically divided into three distinct subsets [Nel23, cp.]:

- Training Subset: This subset is used to train the machine learning model. It consists of data points whose features and labels are used to adjust the model's parameters through optimization.
- Validation Subset: The validation subset has a dual purpose. Firstly, it aids in tuning the model's hyperparameters, which impact the model's behaviour but are not optimized during training. Secondly, it helps to prevent overfitting by indicating when the optimization should stop.
- Test Subset: The test subset remains untouched during the training and validation phases. It is reserved for final model evaluation. Since the model has not encountered these data points during optimization, this subset simulates real-world scenarios, allowing us to assess the model's performance on truly unseen data.

In the case of this study, the dataset undergoes a partitioning process, where 85% is allocated to a training set, 10% is designated for a validation set, and the remaining 5% is assigned to a testing set. The individual structures are not randomly assigned to the three groups but in consideration of their depth. This ensures that each dataset is assigned the full diversity of structures. Undersampling is used for depths that exceed the previously defined maximum of 10,000 structures per depth. In this case, only 10,000 structures of a depth are examined instead of all structures to avoid further skewing of the dataset. The 10,000 structures are selected according to complexity to enable the later model to learn more complex relationships.[Huy22, cp.]

Secondly, in each set, fictitious UBO edges are added to the already existing UBO edges as negative examples, as already described in Chapter 3.2.3. The RandomLinkSplit function of the PyTorch geometric package is used for this. Finally, all positive and negative UBO edges are removed from the datasets and set aside. This step prevents information about the UBOs in the training or validation from being viewed by the model beforehand.

Finally, the ownership structures in the different datasets are transformed from a directed graph into an undirected graph. This transformation seems contradictory since ownership structures primarily concern direction and value declaration. However, undirectedness is vital for the free exchange of messages in the subsequent message passing of the GNN.

Batching in machine learning refers to dividing the training dataset into smaller subsets, or batches, and feeding these batches to the model during training. Batching is used to be more memory efficient and create stable and fast training [Pao20, cp.]. Although PyTorch Geometric offers some functions for batching on graph data [PyT23b, cp.], batching on graphs is way more

complex than batching on images, text or tabular data because graphs have more constraints like the connection between nodes. Batching with the help of these functions works quite well for homogeneous graphs, but the attempt to split the heterogeneous graph into batches failed. Often, ownership structures were torn apart or existing edges led nowhere after the batching. On the other hand, the created dataset with a size of several hundred MB is still manageable for training; therefore, batching was optional.

4.5.2 Model Definition

As pointed out in Chapter 3.2.3, several approaches exist for predicting new links on graphs. Due to the complexity of the heterogeneous graph and the good results of GAEs in other studies, the model architecture of GAE is applied to the problem at hand and further refined.

The encoder takes over the task of representation learning and thus consists of different layers. Due to the inductivity of the problem at hand, only two of the three state-of-the-art layers presented in Chapter 3.2.4 come into question. In this case, it is GraphSAGE and GAT. GraphSAGE layers are promising in this regard, as they are automatically inductive due to their sampling strategy and have been able to put forward a good result in predicting links in some comparisons. GAT layers are again promising since they do not include the edge attributes in their calculations as GraphSAGE or others do. This inclusion can be helpful, especially in the present use case, where edge attributes are more critical than node attributes in predicting UBOs. Furthermore, the attention mechanism has established itself in many areas of deep learning and delivers good results there.

PyTorch Geometric, as the library used to define and train the GML models in this use case, generally deals with homogeneous models. This generally contradicts the homogeneity of the outgoing problem. However, homogeneous models can be transformed into models that can learn effectively on heterogeneous graphs using the approach of Schlichtkrull et al. [Sch+17]. To enable effective message passing between node types and layers, separate layers are first defined for each node type. Depending on the connection types of the individual nodes, these separate layers are connected utilizing interlayers, through which the information can flow from one node type to another. This principle is illustrated in Figure 26. On the left side is a homogeneous network with two layers and subsequent activation functions (ReLU). On the right, the heterogeneous model with two layers is shown for the present graph of people and entities, where

messages flow from top to bottom over the different node types. The model's output is the latent z embeddings for persons and entities.

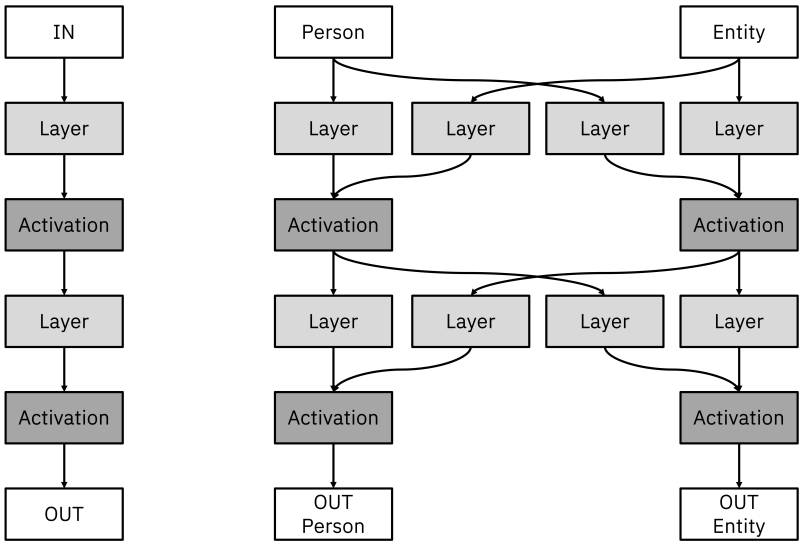

Figure 26: Homogeneous and Heterogeneous Model

The decoder must also be adapted to the given problem. Thus, the standard definition of the decoder assumes a homogeneous graph and multiplies the generated node representations by themselves to reconstruct the graph's adjacency matrix. Due to the graph's heterogeneity and the domain knowledge that UBO edges can only exist between persons and entities, the effort during the multiplication can be reduced. This is possible by multiplying only the representations of persons with those of entities. From the resulting matrix, the probabilities of a UBO edge between each person-entity pair can then be read.

The reconstruction loss between the original and reconstructed graphs is calculated during training. The binary cross entropy is used for this. Reconstruction loss quantifies the dissimilarity between the original graph and its reconstructed version derived from the latent space embeddings generated by the autoencoder. This process serves as a means to ensure that the learned embeddings effectively capture the crucial structural information present in the input graph.[Cia+23, cp.p.14f.]

4.5.3 Model Training and Tuning

The training and the evaluation of the models are executed on a closed instance of Google Colab. This setup provides the necessary hardware resources and

ensures consistent and undisturbed performance. The Google Colab instance has the following hardware configuration:

- Intel(R) Xeon(R) CPU 2.20GHz
- 24 GB of RAM
- Nvidia V100 GPU with 16 GB of Video Random Access Memory (VRAM)

Since it was not feasible to deduce from literature or other sources which is the best architecture for the GAEs encoder, hyperparameter tuning is used to achieve this task within the given time and hardware limitations.

Hyperparameter tuning is a critical process in machine learning model development aimed at optimizing the performance of a model. Hyperparameters govern the behaviour of a machine learning algorithm, but unlike the model's internal parameters, they are not learned from the data during training. Instead, they are set by the developer before training begins. The selection of appropriate hyperparameters significantly influences the model's ability to generalize and make accurate predictions on unseen data. There are many approaches to find the best combination of hyperparameters, including manual, grid and random search.[Bal19, cp.p.3-4]

In a manual search, the researcher selects a set of hyperparameters to assess. While this can lead to swift solutions, prior domain knowledge and familiarity with hyperparameter choices within that domain are necessary. Grid search encompasses all conceivable hyperparameter values, creating an extensive matrix of potential combinations. While reproducible and automatable, it becomes less efficient in higher-dimensional hyperparameter spaces. Grid search methodically explores the entire search space, potentially expending resources on less influential parameter values that might not be vital for a given problem's context.[LL19, cp.p.3] Random search randomly samples from a uniform density within the same hyperparameter space that a grid search would span. This algorithm selects a population sample of hyperparameter values, effectively representing the grid's population. However, random search struggles due to its „non-adaptive nature“, meaning it cannot choose hyperparameter sets based on past performance considerations.[BB12, cp.p.285]

Due to the limited time and predictable long training times of the individual models as well as the given hardware limitations and the small space of hyperparameters, a manual hyperparameter search is conducted to test carefully selected parameter combinations. In this case, tunable parameters are mainly

the selected layer architecture, the number of layers and the size of the representations in the first and last layers. This results in 4 upper categories, which are expanded and examined with the other parameters.

- Models with a shallow architecture (2 layers)
- Models with a midrange architecture (8 layers)
- Models with a deep architecture (16 layers)
- Models with constant variable sizes throughout each layer

Each of those groups is further split into different subgroups, testing different combinations of the layered architecture and the variable size, visible in Table 5 in the appendix.

The learning rate is a crucial hyperparameter in machine learning that significantly impacts the training process and the convergence of models. It represents the step size at which a model adjusts its parameters during optimization. A higher learning rate can lead to faster convergence but risks overshooting the optimal solution, while a lower learning rate increases stability but may slow down convergence.[Wu+19, cp.p.1] Setting the optimal learning rate is a science in itself, covered by multiple studies and tests. The results of the paper published by Wilson and Martinez [WM01] show that the fastest and best learning rate on most of the current deep learning tasks is either 0.1 or 0.01. Their findings are decisive when setting the model training's learning rate, which is set to 0.01. Furthermore, a learning rate schedule is applied over training time. Scheduling is the strategy that involves adjusting the learning rate over time as the model trains. It can help address various challenges encountered during optimization, such as fast convergence at the beginning and fine-tuning near the optimal solution. The learning rate schedule typically starts with a higher value to encourage faster convergence in the initial training phases. As training progresses, the learning rate is gradually reduced.[PYJ20, cp.p.1-3] In the case of this study, the learning rate is reduced by half after 250, 500 and 750 epochs.

One crucial aspect of enhancing machine learning models is utilising loss as a guiding criterion. Loss, in this context, signifies quantifying the difference between predicted and actual values. Throughout the training process, the goal is for the training loss to gradually approach zero, indicating that the model is effectively learning from the data. A similar trend can be observed in the validation loss, mainly when the model performs exceedingly well.[Cia+23, cp.p.14f.] However, there comes a point when the validation loss may cease to decrease and instead start increasing again, which can indicate overfitting.

In this phenomenon, the model becomes too fitted to the training data and struggles to generalise to new and test data.[Yin19, cp.p.1-2]

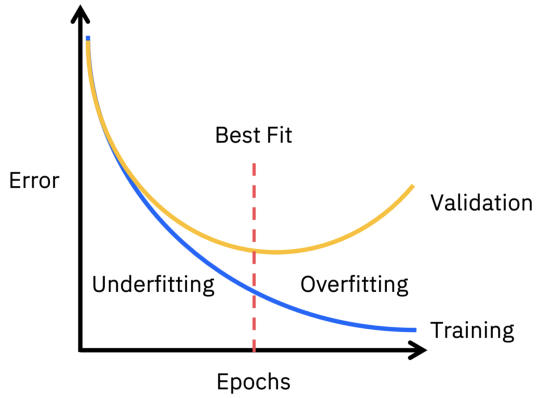

Figure 27: Loss and Early Stopping [Yin19, adapted]

In order to safeguard against the adverse effects of overfitting, a technique known as „Early Stopping“is implemented. Early Stopping protects against overfitting by monitoring the validation loss during training. Implementing Early Stopping involves continuously monitoring the validation loss and comparing it to previous epochs' losses. Suppose the validation loss exceeds a certain threshold or shows a consistent increase for a predefined number of epochs. (See Figure 27) In that case, training is halted, and the model with the lowest validation loss is typically chosen as the final model. This technique balances optimising model performance and preventing overfitting, ultimately contributing to developing more reliable and generalisable machine learning models.[Pre12, cp.p.53-59] Each training session in hyperparameter tuning and all subsequent training sessions are monitored via early stopping. The training is automatically terminated if the validation loss does not improve throughout 25 epochs.

All hyperparameter tuning models are evaluated using the general evaluation metrics defined in Chapter 4.6. The best model architecture is selected based on the calculated metrics on the previously unseen validation data. Cross-validation is applied in the following step to exclude the possibility that the model quality is due to the random selection of the dataset. The most common form of cross-validation is k-fold cross-validation, where the dataset is divided into k equally sized blocks. The model is trained k times, using $k-1$ of those blocks for training and the remaining fold for validation. This process is iterated so each block can serve as the validation set. The calculated model metrics of each fold indicate if the good model performance was by mere chance or due

to good model architecture.[RTL09, cp.p.533-537] In the specific case of this implementation, k is set to three.

Finally, the model is trained with the previously selected architecture and finally evaluated with the evaluation set.

4.6 Evaluation Metrics

Evaluating designed and trained machine learning models is critical in achieving the best problem-solving models. An evaluation of models corresponds to the first step of the demonstration part in the DSR process model. However, the choice of evaluation metrics is vast and needs to be done carefully and with consideration of the model architecture and task. Nevertheless, relying on a single or a limited set of metrics for assessment may only partially capture some performance characteristics, potentially leading to inadequate reflections of model performance.[Bla+20, cp.p1-2]

To evaluate the general ability of the model to predict UBO edges on a graph, the first step is to apply general performance metrics to the model and evaluate them. In the second step, the model is examined with regard to the described problems of the algorithmic approach. For this purpose, the performance of the UBO calculation of both approaches is compared and evaluated, and the behaviour with non-convergent structures is assessed.

4.6.1 General Machine Learning Metrics

The choice of appropriate evaluation metrics depends on the data's nature and the model's objectives, ensuring the performance is accurately assessed for the specific task. For example, in regression tasks, standard evaluation metrics include Mean Squared Error (MSE) or Root Mean Squared Error (RMSE), R^2 to quantify the accuracy of continuous predictions [VC23, cp.p.12]. In contrast, for classification tasks, metrics like accuracy, precision, recall, and F1-score are used to assess the model's ability to classify data into distinct classes [VC23, cp.p.3f.]. Working with other machine learning tasks like processing texts, images, and videos also requires specific metrics [Bla+20, cp.p.15].

As shown in Chapter 3.2.2, although the input formats for graph machine learning models differ from conventional models, their usually do not their primary tasks, such as regression or classification, usually do not differ. For

this reason, the evaluation metrics of the tasks can be directly applied to the evaluation of graph machine-learning models. Since the developed artefact is concerned with the prediction of edges between two nodes, which can be generalized as a binary classification task (0 for no edge, 1 for existing edge), the following section explains metrics for classification and their interpretation possibilities in more detail.

		Predicted Class	
Actual Class		True Positive (TP)	False Positive (FP)
		False Negative (FN)	True Negative (TN)

Table 2: Confusion Matrix

The confusion matrix is a fundamental evaluation tool for binary classification problems, allowing the assessment of a model's performance by categorizing predicted and actual class instances. It consists of True Positive (TP), True Negative (TN), False Positive (FP), and False Negative (FN) values. Various metrics can be derived based on the confusion matrix, such as accuracy, precision, recall and the f1-score, which are widely used for classifier evaluation. In the confusion matrix, the predicted class is represented by the rows, while the columns represent the actual class, as shown in Table 2 .[MM15, cp.p.3] When evaluating a binary classification, the Confusion Matrix indicates how good the model is and where its strengths and weaknesses lie. If the FP value is high, for example, it can be deduced that the model labels many examples as positive that are supposed to be negative.

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \tag{15}$$

Accuracy measures the proportion of correctly classified instances out of the total instances in the dataset, as shown in Equation 15. Accuracy is defined as the ratio of TP and TN to the total number of instances in the dataset. Accuracy can be interpreted as the model's ability to make correct predictions across both positive and negative classes. A high accuracy indicates that the model is making correct predictions most of the time, while a low accuracy suggests that the model's predictions are less reliable.[Dal18, cp.p.48] Balanced accuracy considers the imbalance in class distribution and provides a more balanced measure of a model's ability to make correct predictions across both classes. It is advantageous when one class dominates the dataset, ensuring the evaluation is not overly influenced by the majority class.[GBV20, cp.p.4] The

range of both accuracy and its balanced version is from zero to one, where one indicates a perfect model.

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{16}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{17}$$

Precision represents the ability of a model to identify positive instances out of all instances correctly predicted as positive. It is defined (Equation 16) as the ratio of TP to the sum of TP and FP. On the other hand, recall (Equation 17) quantifies the model's capability to correctly identify positive instances out of all actual positive instances. It is calculated as the ratio of TP to the sum of TP and FN.[Dal18, cp.p.47] Both precision and recall are vital in different scenarios. Precision is crucial when the cost of false positives is high. For instance, in a medical diagnosis system, falsely diagnosing a healthy patient with a disease can lead to unnecessary and potentially harmful treatments. In such cases, high precision is desired to minimize false positives. On the other hand, recall is essential when the cost of false negatives is high. For example, falsely classifying a legitimate email as spam in an email spam filter can lead to missing important messages. In this situation, a high recall is preferred to minimize false negatives.

$$\text{F1 Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{18}$$

The F1 score is a metric that combines precision and recall into a single value, providing a balance between the two. It is a harmonic mean of precision and recall calculated using the following formula in Equation 18. The F1 score ranges between 0 and 1, where 1 indicates perfect precision and recall, while 0 means that either precision or recall is zero. A higher F1 score signifies a better-performing model in terms of both precision and recall.[Bla+20, cp.p.13]

4.6.2 Problem Specific Metrics

The standard metrics for ML models only define the extent to which the model can solve the problem for which it was trained. However, it can only be concluded to a limited degree to what extent the trained model is better or worse than the existing algorithmic approach to determining the UBOs on

ownership structures. For this reason, the trained model is compared against the algorithmic approach in the present edge cases and problem cases. For this purpose, procedures and tests are defined in the following section to compare the two approaches scientifically.

Performance Tests

In retrospect to the research question, it becomes crucial to confirm whether the developed artefact has the potential to address the performance challenges faced by the existing algorithm. Theoretical foundations of algorithmic complexity estimation, typically expressed in Big$\mathcal{O}$ notation, might not be straightforwardly applicable to machine learning models due to the inherent complexity of machine learning algorithms and their dynamic behaviour as they process data [SB22, cp.p.209].

A practical approach addresses this question by implementing a comparative analysis between the proposed artefact and the existing algorithm. A dataset is generated to serve as the basis for evaluation. For each depth of ownership structure, a random sample of 100 structures is extracted. In cases where the desired depth does not yield 100 structures, the available structures are utilised. Notably, the preparation of structures into appropriate formats (Adjacency matrices or PyTorch Geometric Data Objects) is excluded from the comparison and is conducted beforehand. Subsequently, the runtime for each execution is measured. To ensure the comparability of both results, the proposed artefact and the existing algorithm are executed on the given CPU. However, the AI model would also run on the present GPU, and both approaches could be parallelised. The recorded runtimes are then subjected to statistical analysis through hypothesis testing, aiming to assess the temporal performance differences between the two approaches quantitatively.

Hypothesis testing is a fundamental principle in statistical inference, offering a structured framework for making informed decisions about population parameters based on sample data. This process involves comparing two statements: the null hypothesis (H_0), representing a default assumption and the alternative hypothesis (H_1), reflecting the investigated scenario. Through detailed analysis of sample data, the objective is to verify whether sufficient evidence exists to either reject the null hypothesis in favour of the alternative or retain it. The procedure involves systematic steps, including defining the null and alternative hypotheses, selecting an appropriate significance level (alpha), choosing a

suitable statistical test, collecting and evaluating sample data, and finally interpreting the outcomes to draw informed conclusions regarding the population parameter under study.[WDA14, cp.p.372ff.]

The problem at hand requires a difference test, to check whether the two approaches significantly differ in performance. Since the individual results of the measurements can be traced back to a common ownership structure, the normal distribution of the two series can be assumed, and the population's standard deviation is unknown, so a dependent t-student test is conducted [BS13, cp.p.585]. To determine which approach is faster or whether both are equally fast, a two-tailed, a left-tailed and a right-tailed test is performed (Equation 19). The significance level is set at 5% [POD08, cp.p.532] and the degree of freedom is determinded by $n - 1$.

$$\begin{aligned} H_0: &\quad \mu_{\text{diff}} = 0 \\ &\text{-----} \\ H_1: &\quad \mu_{\text{diff}} \neq 0 \\ H_1: &\quad \mu_{\text{diff}} < 0 \\ H_1: &\quad \mu_{\text{diff}} > 0 \end{aligned} \tag{19}$$

The corresponding value for t is calculated with the Equation 20, where $\overline{x}_{\text{diff}}$ is the average of the differences between set A and B, μ_{diff} is the tested μ from the null hypothesis, s is the standard deviation of the differences and n is the size of the sample.

$$t = \frac{\overline{x}_{\text{diff}} - \mu_{\text{diff}}}{s_{\text{diff}}/\sqrt{n}} \tag{20}$$

With the calculated t value and the determined degree of freedom, the p-values are read from the Student-t distribution and finally compared against the determined significance level. The p-value quantifies the likelihood of achieving the observed outcomes under the assumption that the null hypothesis holds. A smaller p-value indicates a higher level of statistical significance for the observed variation. Suppose the p-value exceeds the significance level alpha. In that case, it indicates that the divergence from the null hypothesis lacks statistical importance, and there is no basis to reject the null hypothesis.

Convergence Tests

Furthermore, a comprehensive examination and evaluation of the trained AI model in the context of non-converging ownership structures are necessary to determine its advantages and potential disadvantages in such scenarios. Despite extensive analysis, non-converging structures could not be identified within the present dataset, making it difficult to check the trained model against the described problem.

Constructing an adjacency matrix that does not converge within the existing algorithm is challenging due to the complexities of such structures and their mathematical constraints.

A closer examination of the mathematical theory underlying the concept within the context of an adjacency matrix reveals noteworthy insights. In a directed graph, the adjacency matrix inherently contains zeros along its main diagonal, as entities cannot possess themselves. Additionally, all other matrix elements are less than one and represent the weights of the edges, including capital shares and voting rights expressed as percentages. Furthermore, the matrix is asymmetric due to the directed nature of the graph. Symmetry would imply a reciprocal ownership relationship between companies and individuals, which contradicts established domain knowledge.

Consequently, the adjacency matrix consistently forms a non-symmetric hollow matrix. In mathematical terms, a hollow matrix contains zero values along its main diagonal, indicative of the absence of self-ownership [Gen07, pp. 42, 314].

Considering the potential eigenvalues of such a matrix, the Gershgorin circle theorem becomes relevant for analysis. The Gershgorin circle theorem is a mathematical principle used to estimate the potential range of eigenvalues for a given matrix. Specifically, when applied to a hollow matrix, the theorem states that the magnitudes (absolute values) of the eigenvalues of this matrix will always be equal to or smaller than the sum of the magnitudes of the non-diagonal elements in each row.[LZ19, cp.p.619ff.] As a result, the eigenvalues of such an adjacency matrix can never be greater than 1, which is the primary condition for non-convergence. However, these findings call into question all the concerns of the listed paper regarding the convergence of adjacency matrices.

However, circular ownership structures are considered for further evaluation to examine the issue. Circular ownership structures qualify as test objects since they can be seen as pseudo-non-converging structures, as established

in Chapter 3.1.2. The testing methodology closely resembles that of the performance test, providing a systematic framework for assessment. Ownership structures characterized by circular patterns are prepared as input for the algorithm and the AI model. Subsequently, time measurements are conducted during the calculation process to quantify the computational efficiency of each approach. A comparative analysis is performed through a hypothesis test on the results. This statistical method thoroughly compares the algorithm's and AI model's performance when confronted with circular ownership structures

5 Results

The demonstration and evaluation of the constructed artefact represent a pivotal second to last phase within the DSR process. These phases are executed in accordance with the previously defined problem statements, testing methodologies, and performance metrics. The demonstration phase serves as a comprehensive assessment of its foundational problem-solving capabilities. Within this context, rigorous scrutiny is applied to the outcomes of hyperparameter tuning, explicitly focusing on the optimal GAE architecture and the findings obtained through cross-validation exercises. Subsequently, the evaluation phase focuses on the detailed analysis of results derived from previously defined problem-specific tests. The comprehensive assessment evaluates the artefact's effectiveness in addressing predetermined problems. This assessment encompasses an exploration of the extent to which the trained model effectively mitigates the identified challenges, thereby contributing significantly to the problem-solving objectives.

5.1 Results and Evaluation

In fundamental terms, the presented research has demonstrated that Ownership structures within corporate networks can be effectively represented as heterogeneous graphs. In this representation, entities and individuals become the nodes of the graphs. At the same time, ownership relationships, characterized by attributes such as capital shares, voting rights, and other powers, are represented as edges. Moreover, the UBO identification can be framed as a link prediction problem within this graph-based context. The task involves predicting the link between the target entity for which UBO identification is desired and a person within the ownership structure. To solve this link prediction problem, modern GML models are trained and applied to the graphs at hand. In particular, the GAE architecture can transform irregular and complex graphs with its encoder into structured, meaningful latent encodings, which the decoder uses to reconstruct the graph and especially the UBO edge.

A dataset containing companies, individuals, and ownership structures was employed to train the model adequately. This dataset, originating from the UK, Denmark, Slovakia, and Ukraine ownership registers and collected by the

OpenOwnership initiative, authentically reflects real-world scenarios, containing a substantial 15.5 million entities and individuals connected by 9.1 million ownership relationships. Several essential steps were taken to enhance the dataset's authenticity even further and account for real-world complexities. These involved preprocessing the data as well as augmenting non-beneficial owners, addressing dataset imbalances, and identifying the UBOs using the presented algorithmic method. These efforts resulted in 1.6 million non-trivial and complex ownership structures used to train the GAE model.

Hyperparameter tuning is a critical and intricate phase of model development, primarily because the choice of the optimal model architecture and its parameters is intricately linked to the specifics of the task. Finding the best parameter combination involved a systematic investigation of four distinct main architectures, encompassing a total of 24 carefully hand-chosen configurations that varied in terms of layer types, depth, input and output embedding dimensions. A detailed breakdown of the results is documented in Table 5 in the appendix.

The exploration of these architectures revealed several pivotal insights: Notably, when dealing with low-layer and mid-layer architectures, the distinctions between GraphSAGE and GAT are nuanced, whereby the performance of the individual layers is similar. The low-layer architectures managed to achieve a validation accuracy slightly above 50%, as indicated in Figure 28. However, as the layer count increases within deeper model architectures, a clear trend emerged – deeper models consistently exhibited superior task performance, exceeding their shallower counterparts. The observed improvement in precision with deeper architectures is notable throughout all tests. This finding suggests that deeper models contribute to reducing false positive predictions. Furthermore, the trained models demonstrated a remarkable consistency in recall, consistently exceeding the 90% mark. This high recall rate signifies the model's ability to identify and capture relevant connections within the complex ownership structures, ultimately reducing the risk of overlooking significant UBOs. Moreover, regardless of the depth of the chosen architecture, it could be observed that models with increased output dimensions consistently outperformed their counterparts with lower-dimensional outputs.

In addition to these insights, it is crucial to highlight some unusual and unexpected observations. One such observation is the unexpected superiority of GraphSAGE over GAT in deeper models despite the exclusion of edge attributes within the GraphSAGE architecture. Since GML models, like normal ML models, are basically blackbox models, it cannot be directly deduced why

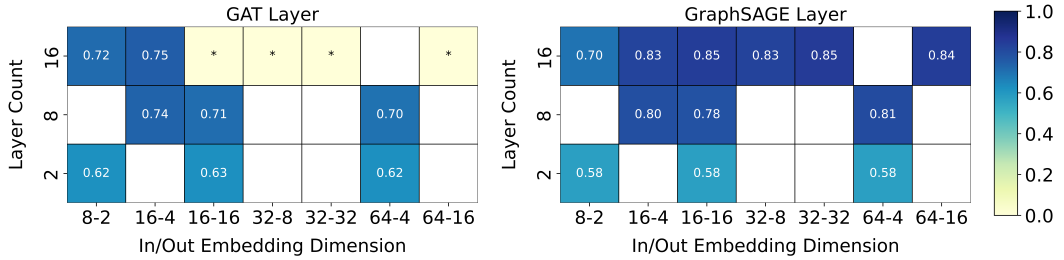

Figure 28: Validation Accuracy of Hyperparameter Combinations

GraphSAGE nevertheless manages to identify the UBOs so well. It is possible that the GraphSAGE model is more oriented towards the structure of ownership graphs if it cannot include edge attributes. Furthermore, the hyperparameter tuning underscored the substantial hardware demands associated with training these models, mainly when dealing with complex encoders. The necessity to keep all variables for backpropagation places considerable computational strain on the training process. Interestingly, GraphSAGE showcased good VRAM efficiency thanks to its learned strategic sampling approach. In contrast, models with GAT layers quickly reached and exceeded the limits of the available 16 GB VRAM, so training more complex GAT models was impossible. Reducing the training data and attention heads did not lead to better results. Due to the given hardware limitations, untrainable models are marked with an asterisk (*) in Table 5 and the Figure 28.

In conclusion, the hyperparameter tuning resulted in identifying the optimal architecture for UBO identification – a deep GraphSAGE model with high embedding dimensions. This configuration proved highly effective, scoring a validation accuracy of 85.2%. Even more impressively, it achieved a validation F1 score of 92.3%, comprising precision at 87.3% and recall at 97.9%.

Cross-validation provides strong evidence that the selected architecture is suitable, reinforcing that the promising results of hyperparameter tuning were not random occurrences. The exemplary training and validation losses in dependence on the progressing training epoch shown in Figure 29 additionally demonstrate the steady improvement of the model during training. Both loss values decrease steadily until an early stopping occurs at epoch 375. From this point onward, only the training loss continues to decrease, whereas the validation loss hardly changes.

Going deeper into the evaluation metrics to assess the trained model's overall link prediction quality gives valuable insights. The validation accuracy of 0.827 indicates that the model correctly identifies approximately 82.7% of UBOs in

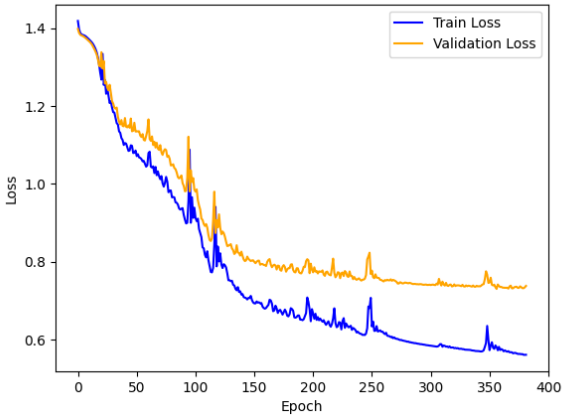

Figure 29: Training and Validation Loss

complex ownership structures. This level of accuracy is adequate, given the complexities of the task. The F1 score, at 0.901, signifies a robust balance between precision and recall. This balance is crucial in UBO identification, as it ensures that a majority of UBOs are correctly identified (precision at 0.843) and that very few UBOs are missed (recall at 0.967). This high recall is particularly important as it minimizes the risk of overlooking potentially critical UBOs within ownership networks.

These metrics affirm that the chosen architecture effectively identifies UBOs within complex ownership structures. These findings align with the fundamental principles of GML and message passing, as shown in the previous literature review. In such complex networks, deep architectures are essential, ensuring that information propagates effectively throughout the entire network. Moreover, the need for high-dimensional embeddings is underscored, allowing the model to capture the multifaceted complexities inherent in graph representations.

The final training, conducted with the selected parameters, marked a crucial stage before the execution of performance and convergence tests. The detailed test results and the corresponding p-values from all hypothesis testing can be found in the appendix in Table 7 and 8.

The comparative analysis of performance yielded several valuable insights. The average computation times can be seen in Figure 30 for both the algorithmic and AI approaches. Notably, the algorithmic approach exhibited exceptional speed in delivering results when dealing with shallower ownership structures. However, as the ownership structure depth increased, the computational time followed an exponential growth pattern, thus verifying the findings described in Chapter 3.1.2. Conversely, the AI model initially showcased relatively higher

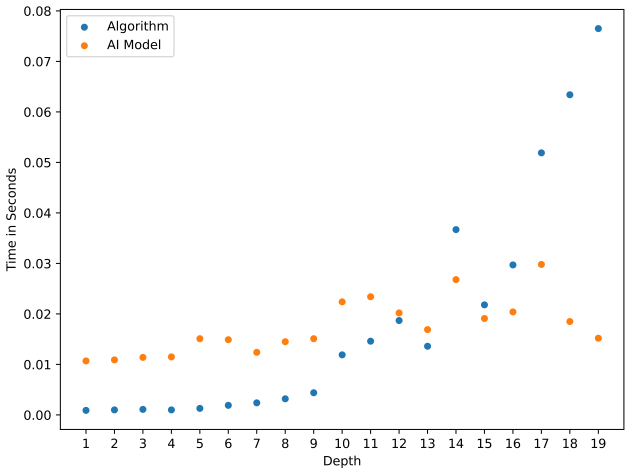

Figure 30: Calculation Mean Time per Depth

runtimes than the algorithmic approach. However, it demonstrated a more linear scaling pattern as the ownership structure's depth increased.

Hypothesis testing results statistically underscored that the AI approach surpassed the algorithmic one regarding runtime efficiency, starting from an ownership depth of 12 to 13, a consistently maintained trend. Remarkably, at depth 12, both approaches converged to statistically equivalent runtimes, with a similar scenario occurring at depth 17 despite a substantial difference in actual runtimes. This discrepancy in the latest case can be attributed to the limited sample size of just two ownership structures.

Further tests involving pseudo-non-convergent circular structures revealed similar, but more evident, trends. Figure 31, illustrating the average computation times for the algorithmic and AI approaches, contrasted the extreme exponential growth in computation times observed with the algorithmic approach. This phenomenon stemmed from the accumulation of matrix powers within circular structures, resulting in endless iterations until the numerical changes in ownership became microscopic and almost indistinguishable, as described in Chapter 3.1.2.

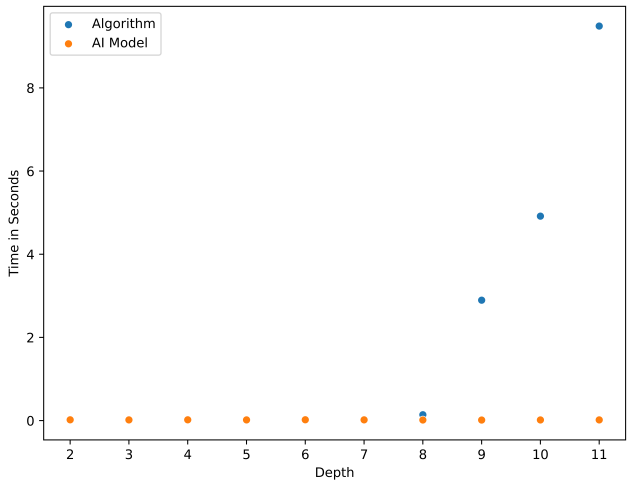

Figure 31: Calculation Mean Time of circular Structures per Depth

In contrast, the AI model consistently demonstrated steady and shorter computation times. Whereas the turning point for normal ownership structures was at a depth of 12, it is already at a depth of 8 for circular structures. Moreover, when examining the accuracy of AI predictions on circular structures, they performed as well as the results obtained through k-fold validation. This suggests that the AI model adeptly handles UBO predictions on circular structures where the algorithmic approach encounters challenges.

In summary, the performance and convergence tests underscore the efficiency of the AI approach, particularly in scenarios involving complex and deep ownership structures.

5.2 Critical Review

Throughout this master's thesis, several relevant domain-specific and technical considerations have emerged, which deserve further exploration and refinement before a perfect system for UBO identification can be achieved.

Firstly, this master's thesis only investigates the numerical ownership attributes defined in the GwG. It is crucial to acknowledge that ownership and exerting influence are two separate domains, as explained by T-rank [T-r19] in their whitepaper. Influence exertion can be considerably more complex than mere ownership attributes. Furthermore, it is essential to recognize that owners often self-reported the data within public ownership registers, which raises questions about data quality. If malicious actors provide false information during data

entry, even the most advanced UBO algorithm would be ineffective. Moreover, most ownership registers are typically limited to specific countries. Particularly in the case of foreign companies, there is a need for a centralized ownership register in order to analyse globally operating companies. Early initiatives are underway to establish global ownership registers, retrieve data from individual registers, and standardize it. However, these efforts are still in their developing stages.[Den22, cp.] [Ope23, cp.]

Regarding the performed hyperparameter tuning, further research could improve some uprising constraints. The results of hyperparameter tuning indicate a substantial dependence on available hardware. The experiment was restricted to testing only 24 carefully selected parameter combinations, which might not provide a comprehensive understanding of the complete hyperparameter space.

Counterintuitively, GraphSAGE performed equally well or outperformed GAT, especially in larger architectures. This contradicts initial expectations because GraphSAGE does not include any Edge Attributes essential for calculating UBOs. This phenomenon suggests that SAGE learned to identify UBOs based on other graph properties and that GAT might not have fully used its tremendous potential. It is also worth noting that four out of the twelve GAT architectures could not be trained due to them exceeding available VRAM. Additionally, the attention mechanism was restricted to a single attention head. In contrast, multi-headed attention, now a standard in various domains, could offer superior performance.[LLH21, cp.p.9] Multi-headed attention allows for more sophisticated functions, enabling each attention head to focus on different input parts.[MLN19, cp.p.1, 8f.] As a result, future research is recommended to undertake a more comprehensive hyperparameter tuning process with superior hardware resources. Moreover, the number of attention heads should be considered as a parameter to evaluate whether better parameter combinations can enhance the model's accuracy. However, it is also essential to balance this goal with the awareness that more complex models may entail longer runtimes, potentially mitigating the current advantages of the existing model.

Overall, the approach of GML is a promising concept. While rapidly expanding, it still needs to be extended in terms of research and especially implementation. Numerous gaps and challenges surfaced during the literature review and artefact development phases. In the realm of PyTorch, there exist many pre-defined layers. However, it is crucial to recognize that out of the 46 pre-established layers, only 12 can effectively utilize edge attributes as input.[PyT23a, cp.] This limitation poses a substantial constraint, considerably reducing the pool

of available layers. Another notable issue is the handling of negative edges introduced by PyTorch to complement positive edges during training. These negative edges are instrumental in training the model effectively. However, PyTorch's implementation presents a random distribution of these negative edges that can lead to a scenario where negative edges are drawn between two contextually independent ownership structures.[PyT23c, cp.] A more advantageous approach would involve a context-aware distribution of negative edges within individual ownership structures. This refinement can significantly enhance the model's performance by aligning the generation of negative edges with the inherent context of the ownership structures under consideration.

6 Conclusion

In conclusion, this master's thesis has introduced a novel approach to address the complex landscape of corporate ownership structures and the identification of their UBOs.

The central premise of this research has been to leverage the inherent interconnectivity within corporate ownership structures, representing them as graphs and using them as input for a graph machine learning model. Identifying UBOs within these structures has been framed as an inductive link prediction problem.

The modern architecture of the GAE was used to solve the given problem. The encoder can convert the ownership graph's complex relationships into latent structured embeddings with the help of the encoder. The decoder can reconstruct and thus predict the actual UBO edges from these embeddings. The conducted hyperparameter tuning has unveiled critical insights for fitting GAE architectures. GAE encoders necessitate a substantial number of layers to facilitate comprehensive message passing within deep ownership structures. Furthermore, the dimensionality of the output embeddings from the encoder must be sufficiently broad to capture the complexities of the graph.

While the trained model exhibited slightly reduced performance in the general identification of UBOs compared to the algorithmic approach, its true strengths became evident in addressing critical challenges inherent to the latter. It showcased significantly improved runtime efficiency when dealing with deep and complex corporate structures. Unlike the algorithmic approach, the AI model exhibited linear runtime scaling with deeper structures compared to the exponential scaling of the algorithm. Notably, the AI model excelled in scenarios where the algorithm failed to provide a performant solution, particularly in pseudo-non-convergence cases within circular ownership structures.

The implications of this research are extensive, with applications spanning various sectors, including regulatory authorities, financial services, banking, and corporate governance. For regulatory authorities and financial institutions, implementing UBO identification models can significantly enhance Anti-Money Laundering compliance efforts, facilitating the detection of suspicious ownership structures and transactions. Banks and lending institutions can employ

these models to assess the risks associated with potential clients, promoting responsible lending and investment practices. Companies can leverage UBO identification models in corporate governance to enhance transparency within their ownership structures, fostering trust among shareholders and stakeholders.

Future research should look into the exploration of more complex patterns of ownership and influence, expand graph machine learning techniques with context-aware positive-negative edge allocation, and conduct more comprehensive and exhaustive hyperparameter tuning to optimize model performance further. These efforts hold the potential to further refine and extend the capabilities of UBO identification models, advancing the cause of transparency, compliance, and risk mitigation while combating financial crime and promoting ethical practices.

Bibliography

[Aga15] Vivek Agarwal. "Research on Data Preprocessing and Categorization Technique for Smartphone Review Analysis". In: *International Journal of Computer Applications* 131.4 (Dec. 17, 2015), pp. 30–36. ISSN: 09758887. DOI: 10.5120/ijca2015907309. (Visited on 08/01/2023).

[Ahm+20] Iftikhar Ahmad et al. "Missing Link Prediction Using Common Neighbor and Centrality Based Parameterized Algorithm". In: *Scientific Reports* 10.1 (Jan. 15, 2020), p. 364. ISSN: 2045-2322. DOI: 10.1038/s41598-019-57304-y. (Visited on 07/20/2023).

[And19] Andres Knobel. *More Beneficial Ownership Loopholes to Plug: Circular Ownership, Fragmented Control and Companies as Parties to the Trust*. Tax Justice Network. Sept. 6, 2019. URL: https://taxjustice.net/2019/09/06/more-beneficial-ownership-loopholes-to-plug-circular-ownership-control-with-little-ownership-and-companies-as-parties-to-the-trust/ (visited on 08/14/2023).

[Ani23] Anil. *Graph Neural Networks*. Wandb.ai. 2023. URL: https://wandb.ai/syllogismos/machine-learning-with-graphs/reports/8-Graph-Neural-Networks--VmlldzozNzcwMTA (visited on 07/24/2023).

[Atz+20] Paolo Atzeni et al. *Weaving Enterprise Knowledge Graphs: The Case of Company Ownership Graphs*. Version 1. OpenProceedings.org, 2020. DOI: 10.5441/002/EDBT.2020.66. (Visited on 01/26/2023).

[Bal19] Balaram Panda. "Hyperparameter Tuning". In: (2019). DOI: 10.13140/RG.2.2.11820.21128. (Visited on 08/15/2023).

[BAY21] Shaked Brody, Uri Alon, and Eran Yahav. "How Attentive Are Graph Attention Networks?" Version 3. In: (2021). DOI: 10.48550/ARXIV.2105.14491. (Visited on 07/25/2023).

[BB12] James Bergstra and Yoshua Bengio. "Random Search for Hyper-Parameter Optimization." In: *Journal of machine learning research* 13.2 (2012).

[Bes+19] Maciej Besta et al. “Demystifying Graph Databases: Analysis and Taxonomy of Data Organization, System Designs, and Graph Queries”. Version 7. In: (2019). DOI: 10.48550/ARXIV.1910.09017. (Visited on 08/01/2023).

[Bis06] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Information Science and Statistics. New York: Springer, 2006. 738 pp. ISBN: 978-0-387-31073-2.

[BKG20] Dor Bank, Noam Koenigstein, and Raja Giryes. “Autoencoders”. Version 2. In: (2020). DOI: 10.48550/ARXIV.2003.05991. (Visited on 07/24/2023).

[BL08] Mike Burkart and Samuel Lee. “One Share - One Vote: The Theory*”. In: *Review of Finance* 12.1 (Jan. 1, 2008), pp. 1–49. ISSN: 1573-692X, 1572-3097. DOI: 10.1093/rof/rfm035. (Visited on 08/24/2023).

[Bla+20] Kathrin Blagec et al. “A Critical Analysis of Metrics Used for Measuring Progress in Artificial Intelligence”. Version 2. In: (2020). DOI: 10.48550/ARXIV.2008.02577. (Visited on 07/19/2023).

[Bon+12] Francesco Bonchi et al. “Fast Matrix Computations for Pairwise and Columnwise Commute Times and Katz Scores”. In: *Internet Mathematics* 8.1-2 (Mar. 2012), pp. 73–112. ISSN: 1542-7951, 1944-9488. DOI: 10.1080/15427951.2012.625256. (Visited on 07/20/2023).

[Bro+17] Michael M. Bronstein et al. “Geometric Deep Learning: Going beyond Euclidean Data”. In: *IEEE Signal Processing Magazine* 34.4 (July 2017), pp. 18–42. ISSN: 1053-5888, 1558-0792. DOI: 10.1109/MSP.2017.2693418. (Visited on 07/18/2023).

[Bru04] Robert F. Bruner. *Applied Mergers and Acquisitions*. Univ. ed. Hoboken, NJ: Wiley, 2004. ISBN: 978-0-471-39534-8.

[BS13] Barbara Illowsky and Susan Dean. *Introductory Statistics*. Texas: openstax, Sept. 19, 2013. 913 pp. ISBN: 978-1-947172-05-0.

[But22] Danny Butvinik. *Graph Analytics for Anti-Money Laundering*. Analytics Vidhya. Feb. 25, 2022. URL: https://medium.com/analytics-vidhya/graph-analytics-for-anti-money-laundering-9d179d60f9d0 (visited on 09/10/2023).

[Cap+21] Cyril Cappi et al. “Dataset Definition Standard (DDS)”. Version 1. In: (2021). DOI: 10.48550/ARXIV.2101.03020. (Visited on 07/28/2023).

[Cha+22] Ines Chami et al. "Machine Learning on Graphs: A Model and Comprehensive Taxonomy". In: *Journal of Machine Learning Research* 23.89 (2022), pp. 1–64. URL: http://jmlr.org/papers/v23/20-852.html.

[Cia+23] Lorenzo Ciampiconi et al. "A Survey and Taxonomy of Loss Functions in Machine Learning". Version 1. In: (2023). DOI: 10.48550/ARXIV.2301.05579. (Visited on 08/16/2023).

[CMA16] Davy Cielen, Arno Meysman, and Mohamed Ali. *Introducing Data Science: Big Data, Machine Learning, and More, Using Python Tools*. Shelter Island, NY: Manning Publications, 2016. 300 pp. ISBN: 978-1-63343-003-7.

[Con+22] Olivia Conjeaud et al. *Anti-Money-Laundering Innovations and Evolving Financial Crime Risks: The Future of Compliance*. McKinsey. May 20, 2022. URL: https://www.mckinsey.com/capabilities/risk-and-resilience/our-insights/anti-money-laundering-innovations-and-evolving-financial-crime-risks-the-future-of-compliance (visited on 09/10/2023).

[Dal18] Hercules Dalianis. "Evaluation Metrics and Evaluation". In: *Clinical Text Mining*. Cham: Springer International Publishing, 2018, pp. 45–53. ISBN: 978-3-319-78502-8. DOI: 10.1007/978-3-319-78503-5_6. (Visited on 07/19/2023).

[DC18] Mirko Di Giacomo and Marisa Cenci. "Corporate Control and Ownership Networks". In: *Corporate Ownership & Control* 15 (Jan. 1, 2018), pp. 86–95. DOI: 10.22495/cocv15i4art8.

[Den22] Denise Lam. *Thomson Reuters Launches Global Beneficial Ownership and Additional Sanctions Data to CLEAR Offering*. Mar. 21, 2022. URL: https://www.thomsonreuters.com/en/press-releases/2022/march/thomson-reuters-launches-global-beneficial-ownership-and-additional-sanctions-data-to-clear-offering.html (visited on 09/11/2023).

[Die17] Reinhard Diestel. *Graph Theory*. 5.th. New York, NY: Springer Berlin Heidelberg, 2017. 447 pp. ISBN: 978-3-662-53621-6.

[Dij23] Bureau van Dijk. *Orbis | Compare Private Company Data | B*. bvd. 2023. URL: https://www.bvdinfo.com/en-gb/our-products/data/international/orbis (visited on 08/04/2023).

[Din+22] Kaize Ding et al. "Data Augmentation for Deep Graph Learning: A Survey". Version 3. In: (2022). DOI: 10.48550/ARXIV.2202.08235. (Visited on 07/26/2023).

[DK00] Dingzhu Du and Ker-I. Ko. *Theory of Computational Complexity*. New York: Wiley, 2000. 491 pp. ISBN: 978-0-471-34506-0.

[DM08] Dan Zuras and Mike Cowlishaw. *IEEE Standard for Floating-Point Arithmetic*. IEEE, Aug. 29, 2008, p. 70. DOI: 10.1109/IEEESTD.2008.4610935. (Visited on 08/15/2023).

[DM22] Debasis Dwibedy and Rakesh Mohanty. "A Note on Hardness of Multiprocessor Scheduling with Scheduling Solution Space Tree". Version 1. In: (2022). DOI: 10.48550/ARXIV.2201.08788. (Visited on 01/25/2023).

[Dom+20] David Dominguez et al. "Panama Papers' Offshoring Network Behavior". In: *Heliyon* 6.6 (June 1, 2020), e04293. ISSN: 2405-8440. DOI: 10.1016/j.heliyon.2020.e04293. (Visited on 08/29/2023).

[DSR11] S.G. Devi, K. Selvam, and S.P. Rajagopalan. "An Abstract to Calculate Big O Factors of Time and Space Complexity of Machine Code". In: *International Conference on Sustainable Energy and Intelligent Systems (SEISCON 2011)*. International Conference on Sustainable Energy and Intelligent Systems (SEISCON 2011). Chennai, India: IET, 2011, pp. 844–847. ISBN: 978-93-80430-00-3. DOI: 10.1049/cp.2011.0483. (Visited on 08/02/2023).

[ER10] Michael Evans and Jeffrey S. Rosenthal. *Probability and Statistics: The Science of Uncertainty*. 2nd ed. New York: W.H. Freeman and Co, 2010. 760 pp. ISBN: 978-1-4292-2462-8.

[EU 22] EU AML/CFT Global Facility. *A History in the Making - Revelations and Evolution in Beneficial Ownership Transparency*. EU AML/CFT Global Facility. June 16, 2022. URL: https://www.global-amlcft.eu/a-history-in-the-making-revelations-and-evolution-in-beneficial-ownership-transparency/ (visited on 08/31/2023).

[FAT18] FATF. *Concealment of Beneficial Ownership*. Egmont Group, 2018. URL: http://www.fatf-gafi.org/publications/methodandtrends/documents/concealment-beneficial-ownership.html.

[FAT23] FATF. *Guidance on Beneficial Ownership for Legal Persons*. Paris: FATF, Mar. 2023, p. 64.

[Fel06] Felix Küsell. "Überblick über Rechtsformen". In: *Praxishandbuch Unternehmensgründung*. Wiesbaden: Gabler, 2006, pp. 340–351. ISBN: 978-3-8349-0165-1. DOI: 10.1007/978-3-8349-9038-9_22. (Visited on 09/10/2023).

[FHL18] George Fletcher, Jan Hidders, and Josep Lluís Larriba-Pey, eds. *Graph Data Management: Fundamental Issues and Recent Developments*. Data-Centric Systems and Applications. Cham: Springer International Publishing, 2018. ISBN: 978-3-319-96192-7. DOI: 10.1007/978-3-319-96193-4. (Visited on 07/17/2023).

[FM97] Julian Franks and Colin Mayer. "Corporate Ownership and Control in the U.k., Germany, and France". In: *Journal of Applied Corporate Finance* 9.4 (1997), pp. 30–45. ISSN: 1745-6622. DOI: 10.1111/j.1745-6622.1997.tb00622.x. (Visited on 09/10/2023).

[Fou23] Clementine Fourrier. *Introduction to Graph Machine Learning*. Hugging Face Blog. Jan. 3, 2023. URL: https://huggingface.co/blog/intro-graphml (visited on 07/18/2023).

[FR22] Francisca Fernando and Richard Berkhout. "Unmasking Control: A Guide to Beneficial Ownership Transparency". In: *Unmasking Control: A Guide to Beneficial Ownership Transparency*. International Monetary Fund, Oct. 7, 2022. ISBN: 9798400208041. URL: https://www.elibrary.imf.org/display/book/9798400208041/9798400208041.xml (visited on 08/22/2023).

[Fu+23] Jun Fu et al. *Variational Disentangled Graph Auto-Encoders for Link Prediction*. June 20, 2023. DOI: 10.48550/arXiv.2306.11315. arXiv: 2306.11315 [cs]. (Visited on 09/21/2023). preprint.

[GBH22] Mikhail Galkin, Max Berrendorf, and Charles Tapley Hoyt. "An Open Challenge for Inductive Link Prediction on Knowledge Graphs". Version 2. In: (2022). DOI: 10.48550/ARXIV.2203.01520. (Visited on 07/18/2023).

[GBV20] Margherita Grandini, Enrico Bagli, and Giorgio Visani. "Metrics for Multi-Class Classification: An Overview". Version 1. In: (2020). DOI: 10.48550/ARXIV.2008.05756. (Visited on 07/19/2023).

[Gen07] James E. Gentle. *Matrix Algebra: Theory, Computations, and Applications in Statistics*. Springer Texts in Statistics. New York, N.Y. ; [London]: Springer, 2007. 528 pp. ISBN: 978-0-387-70872-0.

[GF18] Palash Goyal and Emilio Ferrara. "Graph Embedding Techniques, Applications, and Performance: A Survey". In: *Knowledge-Based Systems* 151 (July 1, 2018), pp. 78–94. ISSN: 0950-7051. DOI: 10.1016/j.knosys.2018.03.022. (Visited on 09/21/2023).

[GH13] Shirley Gregor and Alan R. Hevner. "Positioning and Presenting Design Science Research for Maximum Impact". In: *MIS Quarterly* 37.2 (Feb. 2, 2013), pp. 337–355. ISSN: 02767783, 21629730. DOI: 10.25300/MISQ/2013/37.2.01. (Visited on 07/12/2023).

[Gil+17] Justin Gilmer et al. "Neural Message Passing for Quantum Chemistry". Version 2. In: (2017). DOI: 10.48550/ARXIV.1704.01212. (Visited on 07/24/2023).

[GT02] Michael T. Goodrich and Roberto Tamassia. *Algorithm Design: Foundations, Analysis, and Internet Examples*. New York: Wiley, 2002. 708 pp. ISBN: 978-0-471-38365-9.

[GYZ17] Jonathan L. Gross, Jay Yellen, and Ping Zhang. *Handbook of Graph Theory*. Second edition. Boca Raton: Chapman & Hall/CRC, 2017. ISBN: 978-1-4987-6136-9.

[Han23] Deutsches Handelsregister. *Handelsregister Online*. Handelsregister. 2023. URL: https://www.handelsregister.de/rp_web/information.xhtml (visited on 08/04/2023).

[Hei21] George T. Heineman. *Learning Algorithms: A Programmer's Guide to Writing Better Code*. First edition. Beijing [China] ; Boston [MA]: O'Reilly Media, Inc, 2021. 263 pp. ISBN: 978-1-4920-9106-6.

[Hev+04] Hevner et al. "Design Science in Information Systems Research". In: *MIS Quarterly* 28.1 (2004), p. 75. ISSN: 02767783. DOI: 10.2307/25148625. JSTOR: 10.2307/25148625. (Visited on 07/10/2023).

[HJ23] Gary Hutson and Matt Jackson. *Graph Data Modeling in Python*. S.l.: PACKT PUBLISHING LIMITED, 2023. ISBN: 978-1-80461-803-5.

[HL10] G. Hartman and Open Textbook Library. *Fundamentals of Matrix Algebra*. Online Access: Center for Open Education Open Textbook Library. CreateSpace Independent Publishing Platform, 2010. ISBN: 978-1-4499-4579-4. URL: https://books.google.de/books?id=9Ks9QwAACAAJ.

[HL23] Canran Hou and Huan Liu. "Institutional Cross-Ownership and Stock Price Crash Risk". In: *Research in International Business and Finance* 65 (Apr. 1, 2023), p. 101906. ISSN: 0275-5319. DOI: 10.1016/j.ribaf.2023.101906. (Visited on 08/16/2023).

[Huy22] Chip Huyen. *Designing Machine Learning Systems: An Iterative Process for Production-Ready Applications*. First edition. Sebastopol, CA: O'Reilly Media, Inc, 2022. 367 pp. ISBN: 978-1-09-810796-3.

[HYL18] William L. Hamilton, Rex Ying, and Jure Leskovec. *Inductive Representation Learning on Large Graphs*. Sept. 10, 2018. arXiv: 1706.02216 [cs, stat]. URL: http://arxiv.org/abs/1706.02216 (visited on 06/14/2023). preprint.

[Jam+23] James Manyika et al. *A New Look at How Corporations Impact the Economy and Households | McKinsey*. McKinsey. 2023. URL: https://www.mckinsey.com/capabilities/strategy-and-corporate-finance/our-insights/a-new-look-at-how-corporations-impact-the-economy-and-households#/ (visited on 08/21/2023).

[Jus17] Bundesministerium der Justiz. *Gesetz über das Aufspüren von Gewinnen aus schweren Straftaten (Geldwäschegesetz - GwG)*. June 26, 2017.

[KB16] Jayesh Kudase and Priyanka Bane. "A Brief Study of Graph Data Structure". In: *IJARCCE* 5.6 (June 2016), pp. 268–272. ISSN: 2278-1021. DOI: 10.17148/IJARCCE.2016.5657. (Visited on 01/20/2023).

[KN17] Bhushan Kotnis and Vivi Nastase. "Analysis of the Impact of Negative Sampling on Link Prediction in Knowledge Graphs". Version 2. In: (2017). DOI: 10.48550/ARXIV.1708.06816. (Visited on 07/20/2023).

[Knu97] Donald Ervin Knuth. *The Art of Computer Programming*. 3rd ed. Reading, Mass: Addison-Wesley, 1997. 3 pp. ISBN: 978-0-201-89683-1.

[Kom+16] Matthieu Komorowski et al. "Exploratory Data Analysis". In: Mit Critical Data. *Secondary Analysis of Electronic Health Records*. Cham: Springer International Publishing, 2016, pp. 185–203. ISBN: 978-3-319-43740-8. DOI: 10.1007/978-3-319-43742-2_15. (Visited on 07/28/2023).

[KW16a] Thomas N. Kipf and Max Welling. "Semi-Supervised Classification with Graph Convolutional Networks". Version 4. In: (2016). DOI: 10.48550/ARXIV.1609.02907. (Visited on 07/19/2023).

[KW16b] Thomas N. Kipf and Max Welling. "Variational Graph Auto-Encoders". Version 1. In: (2016). DOI: 10.48550/ARXIV.1611.07308. (Visited on 07/24/2023).

[La 21] Marcello La Rocca. *Advanced Algorithms and Data Structures*. Shelter Island, NY: Manning, 2021. 737 pp. ISBN: 978-1-61729-548-5.

[Lab23] Maxime Labonne. *Hands-on Graph Neural Networks Using Python: Practical Techniques and Architectures for Building Powerful Graph and Deep Learning Apps with PyTorch*. Birmingham, UK: Packt Publishing Ltd., 2023. ISBN: 978-1-80461-070-1.

[Lee+18] Joffrey L. Leevy et al. "A Survey on Addressing High-Class Imbalance in Big Data". In: *Journal of Big Data* 5.1 (Dec. 2018), p. 42. ISSN: 2196-1115. DOI: 10.1186/s40537-018-0151-6. (Visited on 07/26/2023).

[Lex23] LexisNexis. *Ultimate Beneficial Ownership*. LexisNexis. 2023. URL: https://internationalsales.lexisnexis.com/glossary/compliance/ultimate-beneficial-ownership-ubo (visited on 01/25/2023).

[Lia+22] Fan Liang et al. "Survey of Graph Neural Networks and Applications". In: *Wireless Communications and Mobile Computing* 2022 (July 28, 2022). Ed. by Wei Li, pp. 1–18. ISSN: 1530-8677, 1530-8669. DOI: 10.1155/2022/9261537. (Visited on 07/18/2023).

[LL08] Luc Laeven and Ross Levine. "Complex Ownership Structures and Corporate Valuations". In: *Review of Financial Studies* 21.2 (Apr. 2008), pp. 579–604. ISSN: 0893-9454, 1465-7368. DOI: 10.1093/rfs/hhm068. (Visited on 09/10/2023).

[LL19] Petro Liashchynskyi and Pavlo Liashchynskyi. "Grid Search, Random Search, Genetic Algorithm: A Big Comparison for NAS". Version 1. In: (2019). DOI: 10.48550/ARXIV.1912.06059. (Visited on 08/16/2023).

[LLH21] Liyuan Liu, Jialu Liu, and Jiawei Han. *Multi-Head or Single-head? An Empirical Comparison for Transformer Training*. June 17, 2021. DOI: 10.48550/arXiv.2106.09650. arXiv: 2106.09650 [cs]. (Visited on 09/13/2023). preprint.

[Lor19] J Lord. "Open Data Sectors and Communities: Corporate Ownership". In: Zenodo, May 8, 2019. DOI: 10.5281/ZENODO.2677857. (Visited on 07/27/2023).

[Lor23a] Lori Perri. *4 Exciting New Trends in the Gartner Emerging Technologies Hype Cycle*. Gartner. Aug. 23, 2023. URL: https://www.gartner.com/en/articles/what-s-new-in-the-2023-gartner-hype-cycle-for-emerging-technologies (visited on 08/31/2023).

[Lor23b] Lori Perri. *What's New in Artificial Intelligence From the 2023 Gartner Hype Cycle™*. Gartner. Aug. 17, 2023. URL: https://www.gartner.com/en/articles/what-s-new-in-artificial-intelligence-from-the-2023-gartner-hype-cycle (visited on 08/31/2023).

[LRM20] Valliappa Lakshmanan, Sara Robinson, and Michael Munn. *Machine Learning Design Patterns: Solutions to Common Challenges in Data Preparation, Model Building, and MLOps*. First edition. Sebastopol, CA: O'Reilly Media, 2020. 390 pp. ISBN: 978-1-09-811578-4.

[LZ19] Chi-Kwong Li and Fuzhen Zhang. "Eigenvalue Continuity and Gersgorin's Theorem". In: *The Electronic Journal of Linear Algebra* 35 (Dec. 9, 2019), pp. 619–625. ISSN: 1081-3810. DOI: 10.13001/ela.2019.5179. (Visited on 08/24/2023).

[MA22] Firdous Ahmad Mala and Rouf Ali. "The Big-O of Mathematics and Computer Science". In: *Journal of Applied Mathematics and Computation* 6.1 (Jan. 4, 2022), pp. 1–3. ISSN: 25760645, 25760653. DOI: 10.26855/jamc.2022.03.001. (Visited on 08/02/2023).

[MBM22] Venkata Naresh Mandhala, Debnath Bhattacharyya, and Divya Midhunchakkaravarthy. "Need of Mitigating Bias in the Datasets Using Machine Learning Algorithms". In: *2022 International Conference on Advances in Computing, Communication and Applied Informatics (ACCAI)*. 2022 International Conference on Advances in Computing, Communication and Applied Informatics (ACCAI). Chennai, India: IEEE, Jan. 28, 2022, pp. 1–7. ISBN: 978-1-66549-529-5. DOI: 10.1109/ACCAI53970.2022.9752643. (Visited on 07/27/2023).

[Meh+19] Ninareh Mehrabi et al. "A Survey on Bias and Fairness in Machine Learning". Version 3. In: (2019). DOI: 10.48550/ARXIV.1908.09635. (Visited on 07/27/2023).

[MI22] Davide Magnanimi and Michela Iezzi. "Ownership Graphs and Reasoning in Corporate Economics". In: ed. by Ramanath M. et al. Vol. 3135. CEUR Workshop Proceedings. CEUR-WS, 2022. ISBN:

16130073. URL: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85130737202&partnerID=40&md5=c60100198d13b75979e273c9d6b233ab.

[MLN19] Paul Michel, Omer Levy, and Graham Neubig. "Are Sixteen Heads Really Better than One?" Version 3. In: (2019). DOI: 10.48550/ARXIV.1905.10650. (Visited on 09/13/2023).

[MM15] Hossin M and Sulaiman M.N. "A Review on Evaluation Metrics for Data Classification Evaluations". In: *International Journal of Data Mining & Knowledge Management Process* 5.2 (Mar. 31, 2015), pp. 01–11. ISSN: 2231007X, 22309608. DOI: 10.5121/ijdkp.2015.5201. (Visited on 07/19/2023).

[MM22] Alhassan Mumuni and Fuseini Mumuni. "Data Augmentation: A Comprehensive Survey of Modern Approaches". In: *Array* 16 (Dec. 2022), p. 100258. ISSN: 25900056. DOI: 10.1016/j.array.2022.100258. (Visited on 07/26/2023).

[Möl11] Bodo Möller. "Binary Exponentiation". In: *Encyclopedia of Cryptography and Security*. Ed. by Henk C. A. Van Tilborg and Sushil Jajodia. Boston, MA: Springer US, 2011, pp. 84–86. ISBN: 978-1-4419-5905-8. DOI: 10.1007/978-1-4419-5906-5_26. (Visited on 08/03/2023).

[Muh11] Muhammad Mahdi Karim. *African Elephant*. June 20, 2011. URL: https://upload.wikimedia.org/wikipedia/commons/thumb/6/63/African_elephant_warning_raised_trunk.jpg/320px-African_elephant_warning_raised_trunk.jpg.

[Nel23] Hala Nelson. *Essential Math for AI: Next-Level Mathematics for Efficient and Successful AI Systems*. Sebastopol: O'Reilly Media, 2023. 602 pp. ISBN: 978-1-09-810763-5.

[NRG22] NRG-Metrics. *Datasets – NRG Metrics*. 2022. URL: https://nrgmetrics.com/datasets/ (visited on 01/21/2023).

[OB17] Fabiano De S. Oliveira and Valmir C. Barbosa. "A Note on Counting Independent Terms in Asymptotic Expressions of Computational Complexity". In: *Optimization Letters* 11.8 (Dec. 2017), pp. 1757–1765. ISSN: 1862-4472, 1862-4480. DOI: 10.1007/s11590-016-1092-7. (Visited on 08/02/2023).

[Ope17] OpenDataBarometer. *Global Open Data Report*. 4. World Wide Web Foundation, May 2017, p. 36. URL: https://opendatabarometer.org/doc/4thEdition/ODB-4thEdition-GlobalReport.pdf (visited on 08/04/2023).

[Ope23] OpenOwnership. *What We Do*. openownership.org. 2023. URL: https://www.openownership.org/en/about/what-we-do/ (visited on 08/04/2023).

[Öst+10] Hubert Österle et al. "Memorandum zur gestaltungsorientierten Wirtschaftsinformatik". In: *Schmalenbachs Zeitschrift für betriebswirtschaftliche Forschung* 62.6 (Sept. 2010), pp. 664–672. ISSN: 0341-2687, 2366-6153. DOI: 10.1007/BF03372838. (Visited on 01/22/2023).

[Own23] Open Ownership. *Open Ownership Register*. Jan. 20, 2023. URL: https://register.openownership.org/ (visited on 01/20/2023).

[Pan23] Kishan Pandey. *Applications of Graph Data Structure*. Masai School. June 23, 2023. URL: https://www.masaischool.com/blog/applications-of-graph-data-structure/ (visited on 07/17/2023).

[Pao20] Paolo Perrotta. *Programming Machine Learning: From Zero to Deep Learning*. 1st ed. O'Reilly Media, Apr. 14, 2020. 318 pp. ISBN: 1-68050-660-9.

[Par15] European Parliament. *Directive (EU) 2015/849 of the European Parliament and of the Council of 20 May 2015 on the Prevention of the Use of the Financial System for the Purposes of Money Laundering or Terrorist Financing, Amending Regulation (EU) No 648/2012 of the European Parliament and of the Council, and Repealing Directive 2005/60/EC of the European Parliament and of the Council and Commission Directive 2006/70/EC (Text with EEA Relevance)*. May 20, 2015. URL: http://data.europa.eu/eli/dir/2015/849/oj/eng (visited on 01/19/2023).

[PAS14] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. "DeepWalk: Online Learning of Social Representations". In: *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '14: The 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. New York New York USA: ACM, Aug. 24, 2014, pp. 701–710. ISBN: 978-1-4503-2956-9. DOI: 10.1145/2623330.2623732. (Visited on 07/24/2023).

[Pef+07] Ken Peffers et al. "A Design Science Research Methodology for Information Systems Research". In: *Journal of Management Information Systems* 24.3 (Dec. 2007), pp. 45–77. ISSN: 0742-1222,

1557-928X. DOI: 10.2753/MIS0742-1222240302. (Visited on 07/10/2023).

[POD08] Roxy Peck, Chris Olsen, and Jay L. Devore. *Introduction to Statistics and Data Analysis*. 3rd ed. Australia ; Belmont, CA: Thomson Brooks/Cole, 2008. 847 pp. ISBN: 978-0-495-11873-2.

[PPS21] Kirill Polovnikov, Nikita Pospelov, and Dmitriy Skougarevskiy. "Alpha-Indirect Control in Onion-like Networks". Version 2. In: (2021). DOI: 10.48550/ARXIV.2109.07181. (Visited on 01/18/2023).

[Pre12] Lutz Prechelt. "Early Stopping — But When?" In: *Neural Networks: Tricks of the Trade*. Ed. by Grégoire Montavon, Geneviève B. Orr, and Klaus-Robert Müller. Vol. 7700. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 53–67. ISBN: 978-3-642-35288-1. DOI: 10.1007/978-3-642-35289-8_5. (Visited on 07/27/2023).

[Pre99] Bruno R. Preiss. *Data Structures and Algorithms: With Object-Oriented Design Patterns in C++*. Worldwide Series in Computer Science. New York: John Wiley Sons, 1999. 660 pp. ISBN: 978-0-471-24134-8.

[PYJ20] Jieun Park, Dokkyun Yi, and Sangmin Ji. "A Novel Learning Rate Schedule in Optimization for Neural Networks and It's Convergence". In: *Symmetry* 12.4 (Apr. 22, 2020), p. 660. ISSN: 2073-8994. DOI: 10.3390/sym12040660. (Visited on 07/27/2023).

[PyT23a] PyTorch. *GNN Cheatsheet*. 2023. URL: https://pytorch-geometric.readthedocs.io/en/latest/cheatsheet/gnn_cheatsheet.html.

[PyT23b] PyTorch geometric Documentation. *Data Loader*. 2023. URL: https://pytorch-geometric.readthedocs.io/en/latest/modules/loader.html (visited on 08/14/2023).

[PyT23c] PyTorch geometric Documentation. *RandomLinkSplit*. 2023. URL: https://pytorch-geometric.readthedocs.io/en/latest/generated/torch_geometric.transforms.RandomLinkSplit.html (visited on 09/14/2023).

[RD08] Venkateswara Reddy and K Damodaram. "Removal of Duplicates in Database Relations and the Associated Propagation Management". In: *International Journal of Advanced Research in Computer Science* 9.8 (Apr. 208), pp. 767–772. ISSN: 0976-5697. DOI: 10.26483/ijarcs.v9i2.

[Rob11] Jean-Philippe Robé. "The Legal Structure of the Firm". In: *Accounting, Economics, and Law* 1.1 (Jan. 31, 2011). ISSN: 2152-2820. DOI: 10.2202/2152-2820.1001. (Visited on 09/10/2023).

[RTL09] Payam Refaeilzadeh, Lei Tang, and Huan Liu. "Cross-Validation". In: *Encyclopedia of Database Systems*. Ed. by Ling Liu and M. Tamer Özsu. Boston, MA: Springer US, 2009, pp. 532–538. ISBN: 978-0-387-35544-3. DOI: 10.1007/978-0-387-39940-9_565. (Visited on 08/17/2023).

[RWE15] Ian Robinson, James Webber, and Emil Eifrem. *Graph Databases*. Second edition. Beijing: O'Reilly, 2015. 218 pp. ISBN: 978-1-4919-3089-2.

[Sam23] Sameeksha Medewar. *Big O Notation Cheat Sheet | What Is Time & Space Complexity?* Hackr.io. Mar. 22, 2023. URL: https://hackr.io/blog/big-o-notation-cheat-sheet (visited on 08/11/2023).

[San+21] Benjamin Sanchez-Lengeling et al. "A Gentle Introduction to Graph Neural Networks". In: *Distill* 6.8 (Aug. 17, 2021), 10.23915/distill.00033. ISSN: 2476-0757. DOI: 10.23915/distill.00033. (Visited on 07/17/2023).

[SB22] Bhoomi Shah and Hetal Bhavsar. "Time Complexity in Deep Learning Models". In: *Procedia Computer Science*. 4th International Conference on Innovative Data Communication Technology and Application 215 (Jan. 1, 2022), pp. 202–210. ISSN: 1877-0509. DOI: 10.1016/j.procs.2022.12.023. (Visited on 08/18/2023).

[SBH17] Karl Siebertz, David Van Bebber, and Thomas Hochkirchen. "Korrelationsanalyse". In: Karl Siebertz, David Van Bebber, and Thomas Hochkirchen. *Statistische Versuchsplanung*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2017, pp. 381–394. ISBN: 978-3-662-55742-6. DOI: 10.1007/978-3-662-55743-3_11. (Visited on 08/10/2023).

[Sch+17] Michael Schlichtkrull et al. "Modeling Relational Data with Graph Convolutional Networks". Version 4. In: (2017). DOI: 10.48550/ARXIV.1703.06103. (Visited on 07/27/2023).

[SCP20] Stephen Skripak, Anastasia Cortes, and Ron Poff. *Fundamentals of Business, 3rd Edition*. 3rd ed. Pamplin College of Business in Association with Virginia Tech Publishing, Aug. 17, 2020. ISBN: 978-1-949373-87-5. DOI: 10.21061/fundamentals-of-business3e. (Visited on 08/17/2023).

[She18] Andrew J. Sherman. *Mergers and Acquisitions from a to z*. Fourth Edition. New York: AMACOM, American Management Association, 2018. 1 p. ISBN: 978-0-8144-3903-6.

[SK19] Connor Shorten and Taghi M. Khoshgoftaar. "A Survey on Image Data Augmentation for Deep Learning". In: *Journal of Big Data* 6.1 (Dec. 2019), p. 60. ISSN: 2196-1115. DOI: 10.1186/s40537-019-0197-0. (Visited on 07/26/2023).

[SMD21] Claudio Stamile, Aldo Marzullo, and Enrico Deusebio. *Graph Machine Learning: Take Graph Data to the next Level by Applying Machine Learning Techniques and Algorithms*. Birmingham: Packt Publishing, 2021. 319 pp. ISBN: 978-1-80020-449-2.

[SMF20] Saleh Shahinfar, Paul Meek, and Greg Falzon. ""How Many Images Do I Need?" Understanding How Sample Size per Class Affects Deep Learning Model Performance Metrics for Balanced Designs in Autonomous Wildlife Monitoring". In: *Ecological Informatics* 57 (May 2020), p. 101085. ISSN: 15749541. DOI: 10.1016/j.ecoinf.2020.101085. (Visited on 07/28/2023).

[Spi08] Michael Spivak. *Calculus*. Fourth Edition. Houston, Texas: Publish or Perish, 2008. 680 pp. ISBN: 978-0-914098-91-1.

[SS12] Harmanjit Singh and Richa Sharma. "Role of Adjacency Matrix & Adjacency List in Graph Theory". In: *INTERNATIONAL JOURNAL OF COMPUTERS & TECHNOLOGY* 3.1 (Aug. 1, 2012), pp. 179–183. ISSN: 2277-3061. DOI: 10.24297/ijct.v3i1c.2775. (Visited on 07/17/2023).

[Ste20] Steele Blog. *Exploring Circular Ownership in Compliance Investigations | Steele %*. Steele. Dec. 18, 2020. URL: https://steeleglobal.com/exploring-circular-ownership-in-compliance-investigations/ (visited on 08/15/2023).

[Str69] Volker Strassen. "Gaussian Elimination Is Not Optimal". In: *Numerische Mathematik* 13.4 (Aug. 1969), pp. 354–356. ISSN: 0029-599X, 0945-3245. DOI: 10.1007/BF02165411. (Visited on 08/03/2023).

[T-r19] T-rank. *Beneficial Owners Based on Ownership*. T-rank, Nov. 1, 2019. URL: https://docs.trank.no/white_papers/white_paper_beneficial_ownership.pdf.

[Tab+19] Michael A. Tabak et al. "Machine Learning to Classify Animal Species in Camera Trap Images: Applications in Ecology". In: *Methods in Ecology and Evolution* 10.4 (Apr. 2019). Ed. by Theoni Photopoulou, pp. 585–590. ISSN: 2041-210X, 2041-210X. DOI: 10.1111/2041-210X.13120. (Visited on 07/28/2023).

[Tra18] Phi Vu Tran. "Learning to Make Predictions on Graphs with Autoencoders". In: *2018 IEEE 5th International Conference on Data Science and Advanced Analytics (DSAA)*. 2018 IEEE 5th International Conference on Data Science and Advanced Analytics (DSAA). Oct. 2018, pp. 237–245. DOI: 10.1109/DSAA.2018.00034.

[TRM21] Santiago Timón-Reina, Mariano Rincón, and Rafael Martínez-Tomás. "An Overview of Graph Databases and Their Applications in the Biomedical Domain". In: *Database* 2021 (May 18, 2021), baab026. ISSN: 1758-0463. DOI: 10.1093/database/baab026. (Visited on 08/01/2023).

[Tru93] Richard J. Trudeau. *Introduction to Graph Theory*. Dover Books on Advanced Mathematics. New York: Dover Pub, 1993. 209 pp. ISBN: 978-0-486-67870-2.

[Uni22] United Nations Office on Drugs and Crime. *Good Practices and Challenges with Respect to Beneficial Ownership and How It Can Foster and Enhance the Effective Recovery and Return of Proceeds of Crime*. Oct. 15, 2022. URL: https://www.unodc.org/documents/treaties/UNCAC/WorkingGroups/workinggroup2/2022-November-7-11/CAC-COSP-WG.2-2022-CRP.1.pdf.

[vBHM20] Jan vom Brocke, Alan Hevner, and Alexander Maedche. "Introduction to Design Science Research". In: *Design Science Research. Cases*. Ed. by Jan vom Brocke, Alan Hevner, and Alexander Maedche. Cham: Springer International Publishing, 2020, pp. 1–13. ISBN: 978-3-030-46780-7. DOI: 10.1007/978-3-030-46781-4_1. (Visited on 01/18/2023).

[VC23] Gaël Varoquaux and Olivier Colliot. "Evaluating Machine Learning Models and Their Diagnostic Value". In: *Machine Learning for Brain Disorders*. Ed. by Olivier Colliot. Springer, June 2023. URL: https://hal.science/hal-03682454.

[Vel+17] Petar Veličković et al. "Graph Attention Networks". Version 3. In: (2017). DOI: 10.48550/ARXIV.1710.10903. (Visited on 07/19/2023).

[VGB11] Stefania Vitali, James B. Glattfelder, and Stefano Battiston. "The Network of Global Corporate Control". In: *PLOS ONE* 6.10 (Oct. 26, 2011), e25995. ISSN: 1932-6203. DOI: 10.1371/journal.pone.0025995. (Visited on 08/23/2023).

[VK15] Vijay Vaishnavi and William Kuechler. *Design Science Research Methods and Patterns: Innovating Information and Communication Technology*. Second edition. Boca Raton: CRC Press, Taylor & Francis Group, 2015. 373 pp. ISBN: 978-1-4987-1525-6.

[War23] Ian Ward. *JSON Lines*. 2023. URL: https://jsonlines.org/ (visited on 08/01/2023).

[WDA14] Carolyn Warren, Kimberly Denley, and Emily Atchley. *Beginning Statistics*. 2nd edition. Charleston, SC: Hawkes Learning Systems, 2014. 983 pp. ISBN: 978-1-932628-68-5.

[Wen18] Lilian Weng. *From Autoencoder to Beta-VAE*. Aug. 12, 2018. URL: https://lilianweng.github.io/posts/2018-08-12-vae/ (visited on 07/24/2023).

[WF94] Stanley Wasserman and Katherine Faust. *Social Network Analysis: Methods and Applications*. 1st ed. Cambridge University Press, Nov. 25, 1994. ISBN: 978-0-521-38707-1. DOI: 10.1017/CBO9780511815478. (Visited on 09/19/2023).

[Wil+23] Virginia Vassilevska Williams et al. "New Bounds for Matrix Multiplication: From Alpha to Omega". Version 1. In: (2023). DOI: 10.48550/ARXIV.2307.07970. (Visited on 08/03/2023).

[Wil09] Robin J. Wilson. *Introduction to Graph Theory*. 4. ed., [Nachdr.] Harlow Munich: Prentice Hall, 2009. 171 pp. ISBN: 978-0-582-24993-6.

[WM01] D.R. Wilson and T.R. Martinez. "The Need for Small Learning Rates on Large Problems". In: *IJCNN'01. International Joint Conference on Neural Networks. Proceedings (Cat. No.01CH37222)*. International Joint Conference on Neural Networks (IJCNN'01). Vol. 1. Washington, DC, USA: IEEE, 2001, pp. 115–119. ISBN: 978-0-7803-7044-9. DOI: 10.1109/IJCNN.2001.939002. (Visited on 07/27/2023).

[Wri+02] Peter Wright et al. "The Structure of Ownership and Corporate Acquisition Strategies". In: *Strategic Management Journal* 23.1 (Jan. 2002), pp. 41–53. ISSN: 0143-2095, 1097-0266. DOI: 10.1002/smj.208. (Visited on 09/10/2023).

[Wu+19] Yanzhao Wu et al. “Demystifying Learning Rate Policies for High Accuracy Training of Deep Neural Networks”. Version 2. In: (2019). DOI: 10.48550/ARXIV.1908.06477. (Visited on 08/16/2023).

[Wu+22] Haixia Wu et al. “Link Prediction on Complex Networks: An Experimental Survey”. In: *Data Science and Engineering* 7.3 (Sept. 2022), pp. 253–278. ISSN: 2364-1185, 2364-1541. DOI: 10.1007/s41019-022-00188-2. (Visited on 07/24/2023).

[WV21] Xing Wang and Alexander Vinel. “Benchmarking Graph Neural Networks on Link Prediction”. In: (2021). DOI: 10.48550/ARXIV.2102.12557. (Visited on 07/20/2023).

[Xia+21] Feng Xia et al. “Graph Learning: A Survey”. In: *IEEE Transactions on Artificial Intelligence* 2.2 (Apr. 2021), pp. 109–127. ISSN: 2691-4581. DOI: 10.1109/TAI.2021.3076021. (Visited on 07/18/2023).

[YCS16] Zhilin Yang, William W. Cohen, and Ruslan Salakhutdinov. “Revisiting Semi-Supervised Learning with Graph Embeddings”. Version 2. In: (2016). DOI: 10.48550/ARXIV.1603.08861. (Visited on 07/18/2023).

[Yin19] Xue Ying. “An Overview of Overfitting and Its Solutions”. In: *Journal of Physics: Conference Series* 1168 (Feb. 1, 2019), p. 022022. DOI: 10.1088/1742-6596/1168/2/022022.

[Zah16] Lisna Zahrotun. “Comparison Jaccard Similarity, Cosine Similarity and Combined Both of the Data Clustering With Shared Nearest Neighbor Method”. In: *Computer Engineering and Applications Journal* 5.1 (Feb. 29, 2016), pp. 11–18. ISSN: 2252-5459, 2252-4274. DOI: 10.18495/comengapp.v5i1.160. (Visited on 07/20/2023).

[Zai+17] Syed Haider Ali Zaidi et al. “Panama Papers and the Dilemma of Global Financial Transparency”. In: *International Journal of Modern Research in Management* 1.2 (Dec. 2017), pp. 18–35. ISSN: 2522-638X.

[Zha+21] Xiao-Meng Zhang et al. “Graph Neural Networks and Their Current Applications in Bioinformatics”. In: *Frontiers in Genetics* 12 (July 29, 2021), p. 690049. ISSN: 1664-8021. DOI: 10.3389/fgene.2021.690049. (Visited on 07/18/2023).

[Zho+20] Jie Zhou et al. “Graph Neural Networks: A Review of Methods and Applications”. In: *AI Open* 1 (2020), pp. 57–81. ISSN: 26666510. DOI: 10.1016/j.aiopen.2021.01.001. (Visited on 01/26/2023).

[Zho21] Tao Zhou. "Progresses and Challenges in Link Prediction". In: *iScience* 24.11 (Nov. 2021), p. 103217. ISSN: 25890042. DOI: 10.1016/j.isci.2021.103217. (Visited on 07/24/2023).

Appendix

The complete codebase for the developed artefact used in this master's thesis, including the dataset, data preprocessing and augmentation, all AI models, and the implementation for the artefact evaluations, can be found in the following public GitHub repository and also on the attached data medium.

`https://github.com/NiklasUllmann/NAK_MasterThesis`

.1 Literature Review

k	Full Formula	# of Matrix Mults.
1	A	0
2	$A + A \cdot A$	1
3	$A + A \cdot A + A \cdot A \cdot A$	3
4	$A + A \cdot A + A \cdot A \cdot A + A \cdot A \cdot A \cdot A$	6
5	$A^1 + A^2 + A^3 + A^4 + A \cdot A \cdot A \cdot A \cdot A$	10
6	$A^1 + A^2 + A^3 + A^4 + A^5 + A \cdot A \cdot A \cdot A \cdot A \cdot A$	15

Table 3: Necessary Matrix Multiplications for Matrix Power Summation up to k

.2 Artefact Development

Correlation/ p Value	**share**	**voting**	**other_power**
share	1 / 0	0.493 / 0	-0.014 / 0
voting	0.493 / 0	1 / 0	0.428 / 0
other_power	-0.014 / 0	0.428 / 0	1/0

Table 4: Correlation and p Values of Edge Attributes

.3 Results and Evaluation

Architecture	**Layer Type**	**In-Channels**	**Out-Channels**	**Layer**	**Train Acc**	**Train F1**	**Train Prec**	**Train Rec**	**Val Acc**	**Val F1**	**Val Prec**	**Val Rec**	**Epochs**
Low Layer	SAGE	8	2	2	0.565	0.288	0.549	0.195	0.577	0.290	0.563	0.195	51
		64	4	2	0.565	0.289	0.556	0.195	0.577	0.291	0.571	0.195	57
	GAT	8	2	2	0.647	0.824	0.719	0.965	0.624	0.817	0.705	0.972	426
		64	4	2	0.645	0.807	0.708	0.964	0.625	0.817	0.705	0.972	219
Mid Layer	SAGE	16	4	8	0.813	0.892	0.834	0.959	0.795	0.887	0.816	0.972	262
		64	4	8	0.826	0.899	0.853	0.951	0.805	0.890	0.825	0.967	251
	GAT	16	4	8	0.756	0.868	0.814	0.966	0.737	0.864	0.790	0.980	437
		64	4	8	0.721	0.849	0.776	0.937	0.701	0.855	0.775	0.952	199
High Layer	SAGE	8	2	16	0.720	0.841	0.778	**0.996**	0.696	0.833	0.754	**0.999**	209
		16	4	16	0.855	0.924	0.880	0.983	0.827	0.906	0.846	0.987	387
		32	8	16	0.857	0.925	0.881	0.974	0.832	0.908	0.845	0.981	229
		64	16	16	0.884	0.946	0.910	0.985	0.845	0.920	0.867	0.980	312
	GAT	8	2	16	0.728	0.825	0.786	0.943	0.716	0.829	0.773	0.957	456
		16	4	16	0.776	0.872	0.819	0.945	0.752	0.869	0.802	0.949	332
		32	8	16	*	*	*	*	*	*	*	*	*
		64	16	16	*	*	*	*	*	*	*	*	*
Linear Channels	SAGE	16	16	2	0.565	0.289	0.556	0.195	0.578	0.291	0.571	0.195	80
		16	16	8	0.820	0.888	0.847	0.945	0.783	0.870	0.809	0.945	175
		16	16	16	0.881	0.946	0.909	0.986	0.846	**0.924**	0.867	0.988	275
		32	32	16	**0.894**	**0.954**	**0.921**	0.989	**0.852**	0.923	**0.873**	0.979	270
	GAT	16	16	2	0.646	0.823	0.717	0.965	0.626	0.817	0.704	0.973	233
		16	16	8	0.728	0.851	0.792	0.920	0.714	0.843	0.765	0.943	149
		16	16	16	*	*	*	*	*	*	*	*	*
		32	32	16	*	*	*	*	*	*	*	*	*

Table 5: Results and Metrics of Hyperparameter Tuning

k	Train Acc	Train F1	Train Prec	Train Rec		Val Acc	Val F1	Val Prec	Val Rec		Epochs
1	0.888	0.948	0.915	0.982		0.843	0.917	0.864	0.976		351
2	0.864	0.928	0.890	0.972		0.825	0.901	0.850	0.957		215
3	0.839	0.903	0.855	0.955		0.814	0.885	0.815	0.968		141
Avg.	0.864	0.926	0.887	0.970		0.827	0.901	0.843	0.967		235.7

Table 6: Result and Metrics of k-fold Cross Validation

Depth	Algo Mean Time	AI Mean Time	AI Accuracy	Sample Size	p value (two tailed)	Accept H1 Algo != AI	p value (left tailed)	Accept H1 Algo < AI	p value (right tailed)	Accept H1 Algo > AI
1	0.0009	0.0107	0.6	100	0.0000	True	0.0000	True	1.0000	False
2	0.0010	0.0109	0.632	100	0.0000	True	0.0000	True	1.0000	False
3	0.0011	0.0114	0.654	100	0.0000	True	0.0000	True	1.0000	False
4	0.0010	0.0115	0.757	100	0.0000	True	0.0000	True	1.0000	False
5	0.0013	0.0151	0.853	100	0.0000	True	0.0000	True	1.0000	False
6	0.0019	0.0149	0.759	100	0.0000	True	0.0000	True	1.0000	False
7	0.0024	0.0124	0.833	100	0.0000	True	0.0000	True	1.0000	False
8	0.0032	0.0145	0.72	100	0.0000	True	0.0000	True	1.0000	False
9	0.0044	0.0151	0.841	100	0.0000	True	0.0000	True	1.0000	False
10	0.0119	0.0224	0.754	100	0.0000	True	0.0000	True	1.0000	False
11	0.0146	0.0234	0.793	100	0.0000	True	0.0000	True	1.0000	False
12	0.0187	0.0202	0.897	100	0.2720	False	0.1504	False	0.8496	False
13	0.0136	0.0169	0.731	100	0.0013	True	0.0009	True	0.9991	False
14	0.0367	0.0268	0.854	100	0.0001	True	0.9998	False	0.0002	True
15	0.0218	0.0191	0.794	100	0.0030	True	0.9928	False	0.0072	True
16	0.0297	0.0204	0.729	100	0.0000	True	1.0000	False	0.0000	True
17	0.0519	0.0298	0.764	70	0.0000	True	1.0000	False	0.0000	True
18	0.0634	0.0185	0.733	15	0.0000	True	1.0000	False	0.0000	True
19	0.0765	0.0152	1	2	0.0763	False	0.9944	False	0.0056	True

Table 7: Results of Performance Hypothesis Testing

Depth	Algo Mean Time	AI Mean Time	AI Accuracy	Sampel Size	p value (two tailed)	Accept H1 Algo != AI	p value (left tailed)	Accept H1 Algo < AI	p value (right tailed)	Accept H1 Algo > AI
2	0.0117	0.0182	0.927	34	0.0108	True	0.0065	True	0.9935	False
3	0.0126	0.0164	0.869	84	0.0065	True	0.0058	True	0.9942	False
4	0.0081	0.0184	0.793	17	0.0008	True	0.0001	True	0.9999	False
5	0.0086	0.0160	0.895	27	0.0000	True	0.0000	True	1.0000	False
6	0.0127	0.0199	0.819	98	0.0000	True	0.0000	True	1.0000	False
7	0.0146	0.0179	0.735	100	0.0000	True	0.0004	True	0.9996	False
8	0.1408	0.0138	0.841	100	0.0496	True	0.9760	False	0.0240	True
9	2.8951	0.0126	0.827	100	0.0000	True	1.0000	False	0.0000	True
10	4.9156	0.0149	0.778	78	0.0000	True	1.0000	False	0.0000	True
11	9.4864	0.0166	0.714	21	0.0000	True	1.0000	False	0.0000	True

Table 8: Results of Performance Hypothesis Testing with Circular Structures

Author's declaration

Hereby I solemnly declare:

1. that this Master's Thesis, titled *Identifying Ultimate Beneficial Ownership from Complex Corporate Structures using Graph Machine Learning* is entirely the product of my own scholarly work, unless otherwise indicated in the text or references, or acknowledged below;
2. I have indicated the thoughts adopted directly or indirectly from other sources at the appropriate places within the document;
3. this Master's Thesis has not been submitted either in whole or part, for a degree at this or any other university or institution;
4. I have not published this Master's Thesis in the past;
5. the printed version is equivalent to the submitted electronic one.

I am aware that a dishonest declaration will entail legal consequences.

Hamburg, 22.09.2023

Niklas Ullmann