SerialExperimentsOlga

Olga Yakobson

February 4, 2022
Version: My First Draft

LMU LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN

Department of Mathematics,
Informatics and Statistics
Institute of Informatics

Artificial Intelligence and
Machine Learning

Bachelor's Thesis

SerialExperimentsOlga

Olga Yakobson

1. Reviewer Prof. Dr. Eyke Hüllermeier
Institute of Informatics
LMU Munich

2. Reviewer John Doe
Institute of Informatics
LMU Munich

Supervisors Jane Doe and John Smith

February 4, 2022

Olga Yakobson
SerialExperimentsOlga
Bachelor's Thesis, February 4, 2022
Reviewers: Prof. Dr. Eyke Hüllermeier and John Doe
Supervisors: Jane Doe and John Smith

LMU Munich
Department of Mathematics, Informatics and Statistics
Institute of Informatics
Artificial Intelligence and Machine Learning (AIML)
Akademiestraße 7
80799 Munich

Abstract

Abstract (different language)

Acknowledgement

Contents

1 Introduction

The field of ML on graph-structured data has recently become an active topic of research. One reason for this is the wide range of domains and problems that are expressible in terms of graphs.

1.1 Motivation

1.2 Research Questions

1.3 Structure

Chapter 2: Related Work Some text

Chapter 3: Problem Description

Chapter 4: Implementation Some text

Chapter 5: Conclusion Finally, the results of this thesis are summarized and a brief outline of promising directions for future research is given.

2 Related Work

Before describing the problem, and later on the experimental setup, we first

1. Revise three common prediction tasks in graph neural networks (GNNs)

2. Give a general overview of how GNNs organize and process graph structured data

3. We further discuss the relation of messasge passing mechanis to the Weisfeiler-Lehman (WL), an algorithm for inspecting wheather two graphs are isomorph.

4. Give an formal definition and description of two GNN architectures, which will be used in our experiments.

2.1 Prediction Tasks and Typical Problems

Graphs naturally appear in numerous application domains, ranging from social analysis, bioinformatics to computer vision. A Graph $G = (V, E)$, where $V = \{v_1, ..., v_n\}$ is a set of $N = |V|$ nodes and $E \subseteq V \times V$ a set of edges betwen those nodes. The unique capability of graphs enables capturing the structural relations among data, and thus allows to harvest more insights compared to analyzing data in isolation [Zha+19]. Graphs therefore can be seen as a general laguage for describing entities and relationships between those entities. Graph neural networks (GNNs) then organize graph structured data to tackle various prediction and classification tasks. Typically, one is interested in one of the following three tasks:

1. **Link prediction:** Predict whether there are missing links between two nodes e.g., knowledge graph completion

2. **Vertex classification & regression:** Predict a property of a node e.g., categorize online users/items

3. **Graph classification & regression:** Here we are interested in classifying or predicting a continous value for the entire graph e.g., predicting a property of a molecule.

In this work the main focus will be on the latter two, node classification (NC) node regression (NR) and graph classification (GC) graph regression (GR) for small- as well as large-sized graphs.

2.2 Passing Messages in GNNs

Graphs, by nature are unstructured. Vertices in graphs have no natural order and can contatin any type of information. In order for machine learning algorithms to be able to make use of graph structured data, a mechanism is needed to organize them in a suitable way [Zho+20a; HYL17; Zha+19].

Message passing is a mechanism [Xu+19; Zho+20a], which embedds into every node information about it's neighbourhood. This can be done in several ways and one way of classifying a GNN is by looking at the underlying message passing machanism. In this paper we will look at a network, where message passing is done via convolutions (graph convolutional network (GCN)). We will however ocasionally use the more general term message passing, as the separation is rather blurred and message passing is seen as a generalization of other, more specific mechanisms

Formally, message passing in a GNN can be described as using two functions: AGGREGATE and COMBINE. The expressive and representational power of a GNN can then be determined by looking at the concrete functions and thier properties, used to implement aggregation and combination. AGGREGATE mixes in every iteration the hidden representation of the node with the representation of nodes neighbourhood. COMBINE then combines the mixed representation togheter with the representation of the node. Each node uses the information from its neighbors to update its embeddings, thus a natural extension is to use the information to increase the receptive field by performing AGGREGATE and COMBINE multiple times.

$$a_v^k = \text{AGGREGATE}^{(k)}(\{h_u^{(k-1)} : u \in \mathcal{N}_{(v)}\}), h_v^{(k)} = \text{COMBINE}^{(k)}(h_v^{(k-1)}, a_v^{(k)})$$

For graph-level predictions an additional READOUT- operation is used:

$$h_G = \text{READOUT}(\{h_v^{(K)} | v \in G\})$$

One useful type of information, which the message passing framework should be able to capture, is the local graph structure. This can be done by choosing functions with appropriate properties. A more detailed explanation will follow in section 2.2.2. In spatial GNN we make the assumption of the similarity of neighbor nodes. To exploit this spatial similarity, we perform composition by stacking multiple layers togheter and increase the receptive field.

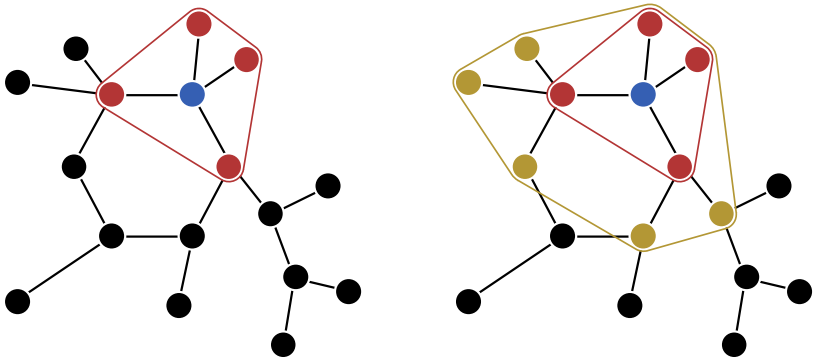

Fig. 2.1.: By performing aggregation k-times we can reach the k-hop neighborhood

2.2.1 Weisfeiler-Lehman Graph Colorings

The Message passing mechanism has a close relation, to the way the Weisfeiler-Lehman (WL) test [WL68] [DMH20] [HV22], an algorithm for deciding wheather two graphs are isomorphic works. Before describing the algorithm, we introduce notations and preliminaries.
Let $G = (V, E, X)$ denote an undirected graph where $V = \{v_1, ..., v_n\}$ is a set of $N = |V|$ nodes and $E \subseteq V \times V$ a set of edges between those nodes. For simplicity we represent an edge v, u by $(v, u) \in E$ or (u, v).$X = [x_1, ..., x_n]^T \in \mathbb{R}^{n \times d}$ is the node feature matrix, where $n = |V|$ is the number of nodes and $x_v \in \mathbb{R}^d$ represents the d-dimensional feature of node v. $\mathcal{N}_v = \{u \in V | (v, u) \in E\}$ is the set of neighboring nodes of node v. A multiset is denoted as $\{\!\{...\}\!\}$ and formally defined as follows.

Definition 2.1 (Multiset)**.** A multiset is a generalized concept of set allowing repeating elements. A multiset X can be formally represented by a 2-tuple as

$X = (S_X, m_X)$, where S_X is the underlying set formed by the distinct elements in the multiset and $m_X : S_X \to \mathbb{Z}^+$ gives the multiplicity (i.e, the number of occurrences) of the elements. If the elements in the multiset are generally drawn from a set X (i.e., $S_X \subseteq X$), then $\mathcal{X}$ is the universe of X and we denote it as $X \subseteq \mathcal{X}$ for ease of notation.

Definition 2.2 (Isomorphism)**.** Two Graphs $\mathcal{G} = (V, E, X)$ and $\mathcal{H} = (P, F, Y)$ are *isomorphic*, denoted as $\mathcal{G} \simeq \mathcal{H}$, if there exists a *bijective* mapping $g : V \to P$ such that $x_v = y_{g(v)}$, $\forall v \in V$ and $(v, u) \in E$ iff $(g(v), g(u)) \in F$. Graph Isomorphism is still an open problem without a known polynomial-time solution.

The 1-dimensional WL algorithm (color refinement)

In the 1-dimensional WL algorithm (1-WL), a label, called color c_v^0 is assigned to each vertex of a graph. Then, in every iteration the colors get updated based on the multiset representation of the neighborhood of the node until convergence. If at some iteration the colorings of the graphs differ, 1-WL decides, that the graphs are not isorprphic.

$$c_v^l \leftarrow \text{HASH}(c_v^{l-1}, \{\!\{c_u^(l-1) | u \in \mathcal{N}_v\}\!\})$$

Algorithmically this can be expressed as follows:

Algorithm 1 1-dim. WL (color refinement)

Input: $G = (V, E, X_V)$
1: $c_v^0 \leftarrow hash(X_v)$ for all $v \in V$
2: **repeat**
3: $\quad c_v^l \leftarrow hash(c_v^{l-1}, \{\!\{c_w^{l-1} : w \in \mathcal{N}_G(v)\}\!\})$ forall $v \in V$
4: **until** $(c_v^l)_{v \in V} = (c_v^{l-1})_{v \in V}$
5: **return** $\{\!\{c_v^l : v \in V\}\!\}$

The 1-WL is a heuristic method, which can efficiently distinguish a broad class of non-isomorphic graphs [BK79]. However there exist some corner cases, where the algorithm fails to classify simple shapes as non-isomorphic.

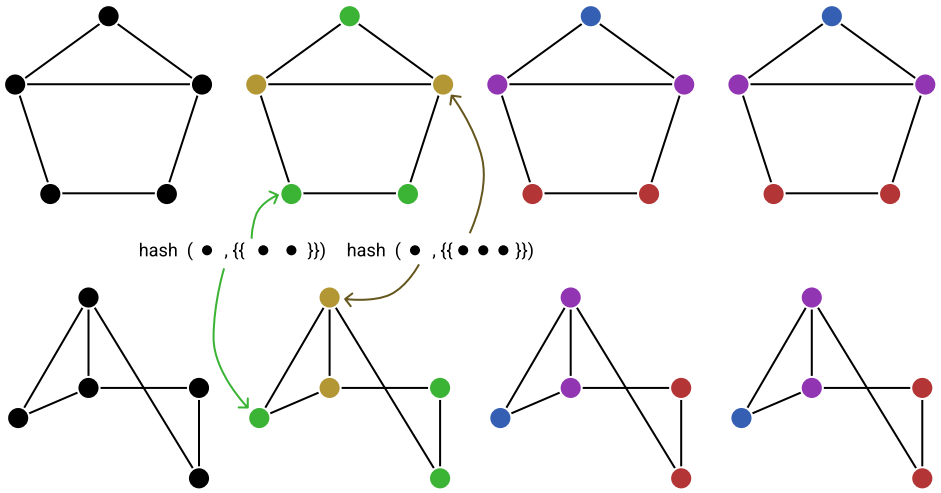

Fig. 2.2.: 1-WLTwo isomorphic graphs. 1-WLassigns same representation

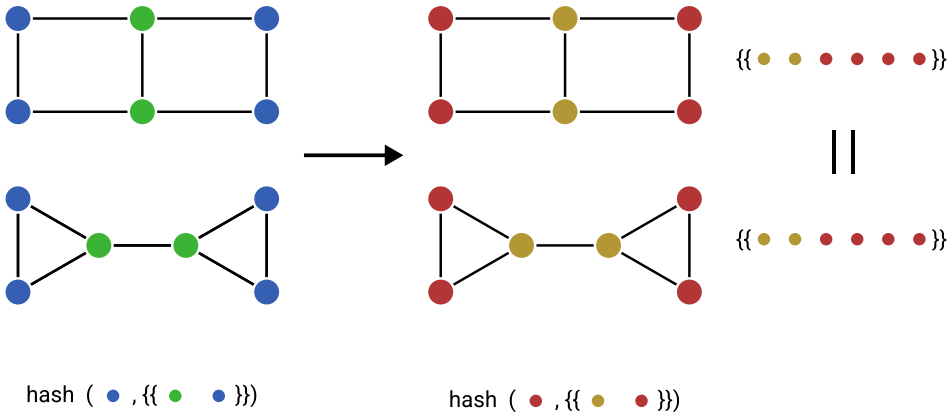

Fig. 2.3.: 1-WL assigned one and the same labeling to two non-isomorphic graphs [LYJ22]

2.2.2 GNN Architectures in this Paper

In the following section we briefly introduce and motivate the choice of two types of networks, which we have chosen to experiementally verify the efficacy of several regularization techniques, which will be discussed in section section 2.2.4.

Since all of GNN encorporate meaasge passing in a way, we decided to chose two interesting architectures for our experiments, which in our view are promising.

Graph Convolutional Network (GCN)

Graph Convolutional Network GCN was originally proposed by Kipf and Welling [KW17] to tackle the problem of semi-supervised node classification, where lables are available for a small subset of nodes. GNNis a simple, but powerful architecture, that scales linearly in the number of graph edges and learns hidden layer representations that encode both local graph structure and features of nodes.

A graph convolutional network (GCN) can formally be expressed via the following layer-wise propagation rule:

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

Where $\tilde{A} = A + I_N$ is the adjacency matrix of the undirected graph $\mathcal{G}$ with added self-connections. I_N is the identity matrix. $\tilde{D}_{ii} = \sum_j \tilde{A}_{ij}$and W^l is a layer-specific trainable weight-matrix.$\sigma(\cdot)$ denotes an activation function, such as $ReLU(\cdot) = max(0, \cdot)$. $H^l \in N \times D$ is the matrix of activations in the l^{th} layer; $H^0 = X$

An application of a two-layer GCN is given by:

$$Z = f(X, A) = \text{softmax}(\hat{A}\ \text{ReLU}(\hat{A}XW^0)W^l)$$

where $\hat{A} = \tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D^{-\frac{1}{2}}}$ is calculated in a preprocessing step. The model uses a single weight matrix per layer and deals with varying node degrees through an appropriate normalization of the adjacency matrix. This model consisted of a 2-layer GCN performed well in a series of experimental tasks, including semi-supervised document classification, semi-superwised node classification in citation networks and semi-supervised entity classification in a bipartite graph extracted from a knowledge graph.The prediction accuracy was evaluated on a set of 1000 examples and additional experiments on deeper models with up to 10 layers have been also provided. GCN outperformed related methods like ManiReg, SemiEmb, LP, DeepWalk, ICA and Platenoid by a significant margin on all of the datasets, which suggests, that the proposed network is capable of encoding both graph structure and node features.

Furthermore it overcame known limitations of existing approaches such as methods based on graph-laplacian regularization, which are limited due to their assumption that edges encode mere similarity of nodes and Skip-gram based methods, that are limited by being based on a multi-step pipeline, which is difficult to optimize.

Overall graph convolutional network (GCN) are widely and successfully used today in many fiels due to thier simplicity and scalability.

Graph Isomorphism Network (GIN)

To overcome the lack of expressivity of popular GNN architectures, Xu et al. designed a new type of GNN, the graph isomorphism network graph isomorphism network (GIN). They proved that GINs are strictly more expressive than a variety of previous GNNs and that they are in fact as powerful as the commonly used Weisfeiler-Lehman graph isomorphism test.

The key idea is to use injective functions, so that the function would never map two different neighbourhoods to the same representations. [Xu+19]

The following layer- wise propagation rule shows a forward pass in a GIN

$$h_v^{(k)} = \mathrm{MLP}^{(k)}(1 + \epsilon^{(k)}) * h_v^{(k-1)} + \sum_{u \in \mathcal{N}(v)} h_u^{(k-1)}$$

$$x = \mathcal{A} : \mathcal{G} \to \mathbb{R}^d$$

Assuming, that the following conditions hold, with a sufficient number of GNN - layer A is as powerful as the WL Test

2.2.3 Weaknesses and Obstacles in GNN Architectures

Because of the way GNNs operate, they tend to suffer from two main obstacles: overfitting and oversmoothing.

Overfitting hinders the generalization ability of a neural network (NN), making it perform poorly on previously unseen data. This problematic occurs expecially when using small datasets, since the model thends to 'memorize' instead of learn the pattern.

Oversmoothing is a condition, where the performance and predictive power of a NN does not imporve of even gets worse when more layers are added. This happens because by stacking multiple layers togheter togheter aggregation is being performed over and over again. This way, the representation of a node is being smoothed - mixed with features of very distant, possibly unrelated nodes. Oversmoothing is a problem mainly for node classification tasks. There is a trade-off between the expressivness of the model (capturing) graph structure by applying multiple layers and oversmoothing, which leads to a model where nodes have the same representation, because they all converge to indistinguishable vectors.[Zho+20b][Has+20] (In spatial GNNs we make the assumption of relatedness by proximity)

A closer examination of underlying causes of oversmoothing was conducted by [Che+20], who suggested, that not message passing itself, but the type of interacting nodes cause this issue. For node classification (NC) tasks, intra-class communication (interaction between two nodes sharing the same class) is useful (signal), whereas inter-class communication (the communication between two nodes sharing different lables) is considered harmful, because it brings interference noise into the feature-representations by mixing unrelated features and therefore making unrelated nodes more similar to each other. Because of that, the the quality of shared information is essencial and should therefore be considered as a benchmark for improvement.

2.2.4 Regularization Techniques

[KGC17] define Regularization as any supplementary technique that aims at making the model generalize better, i.e. produce better results on the test set, which can include various properties of the loss function, the loss optimization algorithm, or other techniques.

One subgroup of regularization is via data, where the training set $\mathcal{D}$ is transformed into a new set $\mathcal{D}_R$ using some stochastic parameter θ, which can be used in various ways, including to manipulate the feature space, create a new, augmented dataset or to change (e.g, thin out the hidden layers of the NN)

In the scope of this work we will look at various regularization techniques and thier efficacy in terms of dealing with the issues of overfitting and oversmoothing The four regularization approaches, as described by [Has+20] are:

DropOut (DO)
DO randomly removes elements of its previous hidden layer $H^{(l)}$ based on independent Bernoulli random draws with a constant success rate at each training iteraration: [Has+20]

$$H^{(l+1)} = \sigma(\mathfrak{R}(A)(Z^{(l)} \odot H^{(l)})W^{(l)})$$

Originally this method was proposed by [Sri+14], who proposed to randomly drop units (along with their connections) from the neural network during training and thus prevent units from co-adapting too much. During training dropout samples from an exponential number of different "thinned" networks.At test time, it is easy to approximate the effect of averaging the predictions of all these thinned networks by simply using a single unthinned network that has smaller weights. This signifcantly reduces overftting and gives major improvements over other regularization methods.

Exspecially when working with small datasets we have much noize, which then leads the model to overfit. DropEdge (DE)

$$H^{(l+1)} = \sigma(\mathfrak{R}(A \odot Z^{(l)})H^{(l)}W^{(l)})$$

NodeSampling (NS)

$$H^{(l+1)} = \sigma(\mathfrak{R}(A)diag(z^{(l)})H^{(l)}W^{(l)})$$

GraphDropConnect (GDC)

$$H^{(l+1)}[:,j] = \sigma(\sum_{i=1}^{f_t} \mathfrak{R}(A \odot Z^{(l)}_{i,j})H^{(l)}[:,i]W^{(l)}[i,j])$$

2.3 Conclusion

GNNs are widely used. They make use of a mechanism called message passing, which is done specifically by using two functions AGGREGATE and COMBINE. The concrete choice of these functions determines the type of GNN and it's expressive power. If we want the network to have the same expressive and representational as the WL, the functions need to be injective in order to map different neighbourhoods to different representations.

Also, since graphs have no natural order, the functions need to be permutation invariant. There are three typeical prediction tasks in GNNs, two of which we will concider in this work. (We focus on node classification regression as well as graph classification regression)

3 Problem Description

[Has+20]

4 Implementation

This section provides a brief overview of experimental setup as well as used libraries and frameworks and gives an explanation for the choices. Despite GNNs being such a big deal and widely used in various domains, there is a lack of standartisation in machine learning on graphs. Tensorflow has no build-in structure for graph representation and expects the input to be tensors or dictionaries. A few attempts were made towards dealing with graph-structured data in a standardized way Spektral is Despite graph neural networks (GNNs) being a hot topic, there still is no standardized way of dealing with graphs in terms of representation, ...

A few efforts have been made to create standardized frameworks and libraries. Such examples are Spektral, and NetworX

4.1 Scope and Limitations

4.2 Experimental Setup

4.3 Evaluation

5 Conclusion

5.1 Future Work

Another proposed regularization technique https://arxiv.org/pdf/2106.07971.pdf

Appendix A

Bibliography

[BK79] László Babai and Ludek Kucera. “Canonical Labelling of Graphs in Linear Average Time”. In: *20th Annual Symposium on Foundations of Computer Science, San Juan, Puerto Rico, 29-31 October 1979*. IEEE Computer Society, 1979, pp. 39–46 (cit. on p. 6).

[Che+20] Deli Chen, Yankai Lin, Wei Li, et al. “Measuring and Relieving the Over-Smoothing Problem for Graph Neural Networks from the Topological View”. In: *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*. AAAI Press, 2020, pp. 3438–3445 (cit. on p. 10).

[DMH20] Clemens Damke, Vitalik Melnikov, and Eyke Hüllermeier. “A Novel Higher-order Weisfeiler-Lehman Graph Convolution”. In: *Proceedings of The 12th Asian Conference on Machine Learning, ACML 2020, 18-20 November 2020, Bangkok, Thailand*. Ed. by Sinno Jialin Pan and Masashi Sugiyama. Vol. 129. Proceedings of Machine Learning Research. PMLR, 2020, pp. 49–64 (cit. on p. 5).

[HYL17] William L. Hamilton, Rex Ying, and Jure Leskovec. “Representation Learning on Graphs: Methods and Applications”. In: *IEEE Data Eng. Bull.* 40.3 (2017), pp. 52–74 (cit. on p. 4).

[Has+20] Arman Hasanzadeh, Ehsan Hajiramezanali, Shahin Boluki, et al. “Bayesian Graph Neural Networks with Adaptive Connection Sampling”. In: *CoRR* abs/2006.04064 (2020). arXiv: 2006.04064 (cit. on pp. 10, 11, 13).

[HV22] Ningyuan Huang and Soledad Villar. “A Short Tutorial on The Weisfeiler-Lehman Test And Its Variants”. In: *CoRR* abs/2201.07083 (2022). arXiv: 2201.07083 (cit. on p. 5).

[KW17] Thomas N. Kipf and Max Welling. “Semi-Supervised Classification with Graph Convolutional Networks”. In: *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017 (cit. on p. 8).

[KGC17] Jan Kukacka, Vladimir Golkov, and Daniel Cremers. “Regularization for Deep Learning: A Taxonomy”. In: *CoRR* abs/1710.10686 (2017). arXiv: 1710.10686 (cit. on p. 10).

[LYJ22] Meng Liu, Haiyang Yu, and Shuiwang Ji. “Your Neighbors Are Communicating: Towards Powerful and Scalable Graph Neural Networks”. In: *CoRR* abs/2206.02059 (2022). arXiv: 2206.02059 (cit. on p. 7).

[Sri+14] Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. “Dropout: a simple way to prevent neural networks from overfitting”. In: *J. Mach. Learn. Res.* 15.1 (2014), pp. 1929–1958 (cit. on p. 11).

[WL68] Boris Weisfeiler and Andrei A. Lehman. “A reduction of a graph to a canonical form and an algebra arising during this reduction”. In: *Nauchno-Technicheskaya Informatsia* 2.9 (1968), pp. 12–16 (cit. on p. 5).

[Xu+19] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. “How Powerful are Graph Neural Networks?” In: *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019 (cit. on pp. 4, 9).

[Zha+19] Si Zhang, Hanghang Tong, Jiejun Xu, and Ross Maciejewski. “Graph Convolutional Networks: A Comprehensive Revew”. In: *Computational Social Networks* (2019) (cit. on pp. 3, 4).

[Zho+20a] Jie Zhou, Ganqu Cui, Shengding Hu, et al. “Graph neural networks: A review of methods and applications”. In: *AI Open* 1 (2020), pp. 57–81 (cit. on p. 4).

[Zho+20b] Kaixiong Zhou, Xiao Huang, Yuening Li, et al. “Towards Deeper Graph Neural Networks with Differentiable Group Normalization”. In: *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*. Ed. by Hugo Larochelle, Marc’Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin. 2020 (cit. on p. 10).

List of Figures

List of Tables

Colophon

This thesis was typeset with LaTeX 2_ε. It uses the *Clean Thesis* style developed by Ricardo Langner. The design of the *Clean Thesis* style is inspired by user guide documents from Apple Inc.

Download the *Clean Thesis* style at `http://cleanthesis.der-ric.de/`.

Declaration

Ich, Olga Yakobson (Matrikel-Nr. 11591478), versichere, dass ich die Masterarbeit mit dem Thema SerialExperimentsOlga selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe. Die Stellen der Arbeit, die ich anderen Werken dem Wortlaut oder dem Sinn nach entnommen habe, wurden in jedem Fall unter Angabe der Quellen der Entlehnung kenntlich gemacht. Das Gleiche gilt auch für Tabellen, Skizzen, Zeichnungen, bildliche Darstellungen usw. Die Bachelorarbeit habe ich nicht, auch nicht auszugsweise, für eine andere abgeschlossene Prüfung angefertigt. Auf § 63 Abs. 5 HZG wird hingewiesen. München, 1. Februar 2023

Munich, February 4, 2022

Olga Yakobson