TECHNISCHE UNIVERSITÄT WIEN

DIPLOMARBEIT

Anomaly Detection in Power Grids by Means of Graph Convolutional Neural Networks

zur Erlangung des akademischen Grades

Diplom-Ingenieur

im Rahmen des Studiums

Physikalische Energie- und Messtechnik

eingereicht von

Wolfgang Lubowski, BSc

Matrikelnummer 1226645

ausgeführt am Institut für Telekommunikation
der Fakultät für Elektrotechnik der Technischen Universität Wien
in Zusammenarbeit mit dem Institut für Angewandte Physik
der Fakultät für Physik der Technischen Universität Wien

Betreuung
Hauptbetreuer: Ao.Univ.Prof. Dipl.-Ing. Dr.techn. Gerald Matz
Mitwirkung: Univ.Ass. Dipl.-Ing. Thomas Dittrich, BSc
Mitbetreuer: Ao.Univ.Prof. Dipl.-Ing. Dr.techn. Martin Gröschl

Wien, 22. Dezember 2022

(Unterschrift Verfasser) (Unterschrift Betreuer)

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Abstract

In recent years, great successes have been achieved through the application of artificial neural networks in engineering and science. In particular, convolutional neural networks (CNNs), i.e., those based on the convolution operation, proved to be useful. However, to allow the application of CNNs, the data is required to have strict neighborhood relationships. This means, the application of CNNs has so far mostly been limited to data acquired in Euclidean domains, such as pixels in a square grid or time series data with constant sampling frequency. For numerous technical and scientific applications the underlying structure is of a more irregular nature, which means that exactly this requirement cannot be met. An example of a problem that is poorly handled by current machine learning technologies is anomaly detection in power grids.

Graphs, that can be used to model arbitrary neighborhood relationships between data points, are often the most natural approach when processing such data. In the field of graph signal processing there is a generalization of the convolution operation for graph signals. Based on this, the new concept of graph convolutional neural networks has been developed over the last years. The applicability of graph convolution neural networks to concrete engineering problems has been insufficiently studied.

This thesis presents an approach, how an anomaly detection model based on a graph convolutional neural network can be implemented. An adaption of the popular GCN framework is proposed, which overcomes the problem of over-smoothing in this network. The performance of the model is then examined for the detection of anomalies in the voltage data from a power grid.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Kurzfassung

In den letzten Jahren wurden durch die Anwendung von künstlichen neuronalen Netzen in Technik und Wissenschaft große Erfolge erzielt. Insbesondere Convolutional Neural Networks (CNNs), d.h. solche, die auf der Faltungsoperation basieren, haben sich als nützlich erwiesen. Um die Anwendung von CNNs zu ermöglichen, müssen die Daten jedoch strenge Nachbarschaftsbeziehungen aufweisen. Das bedeutet, dass die Anwendung von CNNs bisher meist auf Daten beschränkt war, die in einer euklidischen Domäne erfasst wurden, wie z.B. Pixel in einem quadratischen Gitter oder Zeitreihendaten mit konstanter Abtastfrequenz. Bei zahlreichen technischen und wissenschaftlichen Anwendungen ist die zugrundeliegende Struktur unregelmäßigerer Natur, so dass genau diese Anforderung nicht erfüllt werden kann. Ein Beispiel für ein Problem, das von den derzeitigen Technologien des maschinellen Lernens nur unzureichend beherrscht wird, ist die Erkennung von Anomalien in Stromnetzen.

Graphen, mit denen sich beliebige Nachbarschaftsbeziehungen zwischen Datenpunkten modellieren lassen, sind oft der natürlichste Ansatz für die Verarbeitung solcher Daten. Auf dem Gebiet der Graphsignalverarbeitung gibt es eine Verallgemeinerung der Faltungsoperation für Graphsignale. Auf dieser Grundlage wurde in den letzten Jahren das neue Konzept der Graph Convolutional Neural Networks entwickelt. Die Anwendbarkeit von neuronalen Netzen mit Graphenfaltung auf konkrete ingenieurwissenschaftliche Problemstellungen ist bisher nur unzureichend untersucht worden.

In dieser Arbeit wird ein Ansatz vorgestellt, wie ein Modell zur Erkennung von Anomalien auf der Basis eines Graph Convolutional Neural Network implementiert werden kann. Es wird eine Adaption des populären GCN-Frameworks vorgeschlagen, die das Problem des Oversmoothing in diesem Netzwerk behebt. Die Performanz des Modells wird dann für die Erkennung von Anomalien in den Daten der Spannung eines Stromnetzes untersucht.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Danksagung

Ich möchte mich in besonderer Art und Weise für die sehr hilfreiche und geduldige Betreuung dieser Arbeit durch Thomas Dittrich und Prof. Gerald Matz bedanken. In zahllosen ausführlichen Gesprächen haben sie mich dabei unterstützt ein solides Verständnis für die Fragestellungen dieses Themenfeldes zu entwickeln und Lösungen für aufgetretene Probleme zu finden. Bei Prof. Martin Gröschl möchte ich mich besonders dafür bedanken, dass er das interdisziplinäre Zustande kommen dieser Arbeit ermöglicht hat. Besonderer Danke gilt meiner Freundin Anna Kirchmair sowie meinen Eltern Eva und Gerhard Lubowski, dass sie mich in vielfältiger Weise unterstützt haben, den langwierigen Entstehungprozess dieser Arbeit zu einem Ende zu bringen.

Contents

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

List of Symbols

Symbol	Meaning
$\mathcal{G}$	a graph
$\mathcal{V}$	set of nodes in a graph $\mathcal{G}$
$\mathcal{E}$	set of edges in a graph $\mathcal{G}$
N	number of nodes $\lvert\mathcal{V}\rvert$ in a graph
$\mathbf{A}$	weighted adjacency matrix
a_{ij}	weight of the edge connecting the nodes i and j
$\mathbf{x}, \hat{\mathbf{x}}$	univariate signal on a graph and its graph Fourier transform
$\mathbf{X}$	multivariate signal on a graph
$\mathbf{D}$	degree matrix
$\mathbf{L}, \bar{\mathbf{L}}$	unnormalized and normalized graph Laplacian
$\mathcal{N}_i$	the neighborhood of a node i
$\lambda_k, \mathbf{u}_k$	k^{th} eigenvalue and eigenvector of $\mathbf{L}$
$\mathbf{\Lambda}, \mathbf{U}$	matrix representation of eigendecomposition of $\mathbf{L}$
$\mathbf{G}$	a (graph) shift operator
$g(\mathbf{A})$	a polynomial filter; polynomial in $\mathbf{A}$
$\sigma(\cdot)$	a non-linear activation function
θ_k	trainable filter coefficient for the k^{th} order of the polynomial filter
$\mathbf{\Theta}_k$	diagonal matrix of trainable filter coefficients for the k^{th} order of the polynomial filter
g_θ	a diagonal matrix of trainable filter coefficients for the k^{th} order of the polynomial filter (with a slight difference to $\mathbf{\Theta}_k$)
T_k	k^{th} Chebyshev polynomial
$\mathbf{H}^l, \mathbf{H}^{l+1}$	input of output of graph convolutional

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

	neural network layer
$\mathbf{W}^l$	trainable weights in the l^{th} layer of a graph convolutional neural network
$\mathbf{Z}$	output of approximated graph convolution

Table 1: Overview of used symbols.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Chapter 1

Introduction

1.1 Motivation

Power grids are a backbone of our society, ensuring their stability is of utmost importance. This thesis proposes a new method for detecting false data injection attacks in power grids.

Further, central motivation of this thesis is to give a better understanding of the applicability of convolutional neural networks for data acquired in non-Euclidean domains. Graph convolutional neural networks have evolved dynamically in the very recent past. Both major neural network implementation frameworks now provide functionality for graph neural networks: PyTorch offers now *PyTorch Geometric* [1] [2] and TensorFlow offers now *Tensorflow GNN* [3] [4].

Despite the dynamic development of this new field, there are still relatively few examples in the literature of concrete implementations in science or engineering. In particular, very few examples of applications in anomaly detection can be found.

1.2 Problem formulation and contribution

The goal of this thesis is to correctly detect anomalies in voltage measurements from a power grid using a Graph Neural Network generated by False Data Injection.

This thesis makes three main contributions:

- It shows how data from a power grid can be analyzed with a graph neural network. For this purpose, the power grid is represented by a graph and the collected data form the graph signals.
- Furthermore, it is shown how an anomaly detection can be implemented with a convolutional graph neural network. To stay close to existing examples in

the literature, a classification problem is implemented.

- Finally - and this is probably the most important contribution of this thesis - an adaptation of the popular GCN framework is proposed, which solves the problem of over-smoothing in this network. This is achieved by adding an additional term to the propagation rule of the GCN layer, which generates an output signal in case of strong variations of the input signal in the node domain.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

Chapter 2

Anomaly detection in power grids

Power grids are among the largest and most important examples of networks in technology or nature. Their importance for our modern civilization can hardly be overestimated. Measures to enable safe and stable operation are of utmost importance. Large-scale power outages would have a dramatic impact within a relatively short period of time.

2.1 Power grids as graphs

Power grids can be represented in a natural way as graphs: The transmission lines are represented as edges of a graph, the buses as nodes.

Data acquired with Phasor Measurement Units (PMUs) (cf. Section 2.2) are understood as graph signals. Measurements of the voltage at the buses correspond to graph signals at the nodes. Measurements of the currents in the transmission lines could be represented as signals of the edges of the graph. [5]

2.2 Phasor measurement units

The graph signals used for experiments in this thesis were recorded with PMUs. Therefore, a brief introduction to these measurement units is given below.

PMUs are used for continuous monitoring of power grids. For current and voltage, the complex amplitude, i.e. magnitude and phase, is measured. The units are typically distributed throughout the power network, buses and transmission lines should be equipped. To ensure time synchronization between units, a time signal from GPS (or other GNSS) is used.

In some cases, the sampling rate of modern PMUs is high enough to track the exact waveform of current and voltage, allowing for accurate analysis of potential

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

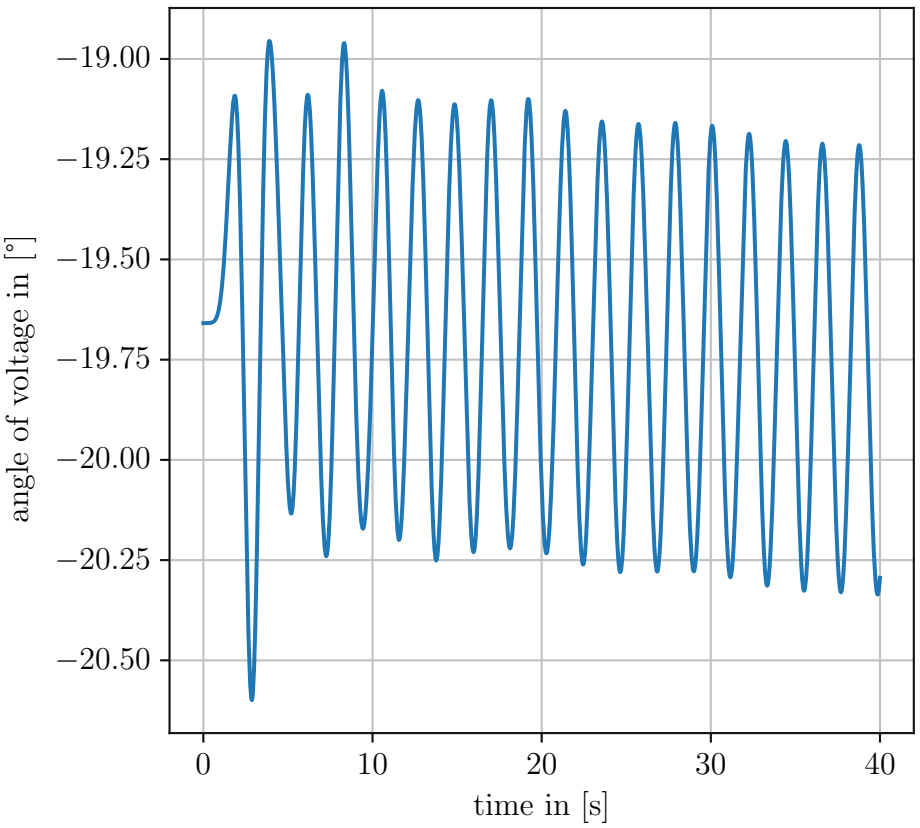

Figure 2.1: Angle of voltage during a poorly oscillation event. Data is taken from [8].

problems [6]. [7] explains that in other cases, a multiple of the grid frequency is used as the sampling frequency, so that only the phase angle progression is recorded, but not the exact waveform, e.g. [8].

Fig. 2.1 and Fig. 2.2 show examples of voltage angle and voltage magnitude recorded with PMUs. The data is from an American grid with 60 Hz nominal grid frequency, and 30 Hz was used as the sampling frequency. The choice of this sampling frequency means that although the exact waveform of the signal cannot be traced, the course of the phase can. The plotted section of 40 seconds shows an oscillation event in the grid. Due to a fault case a badly damped oscillation of grid frequency, voltages and currents has occurred. The example is taken from the test case library of [8].

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

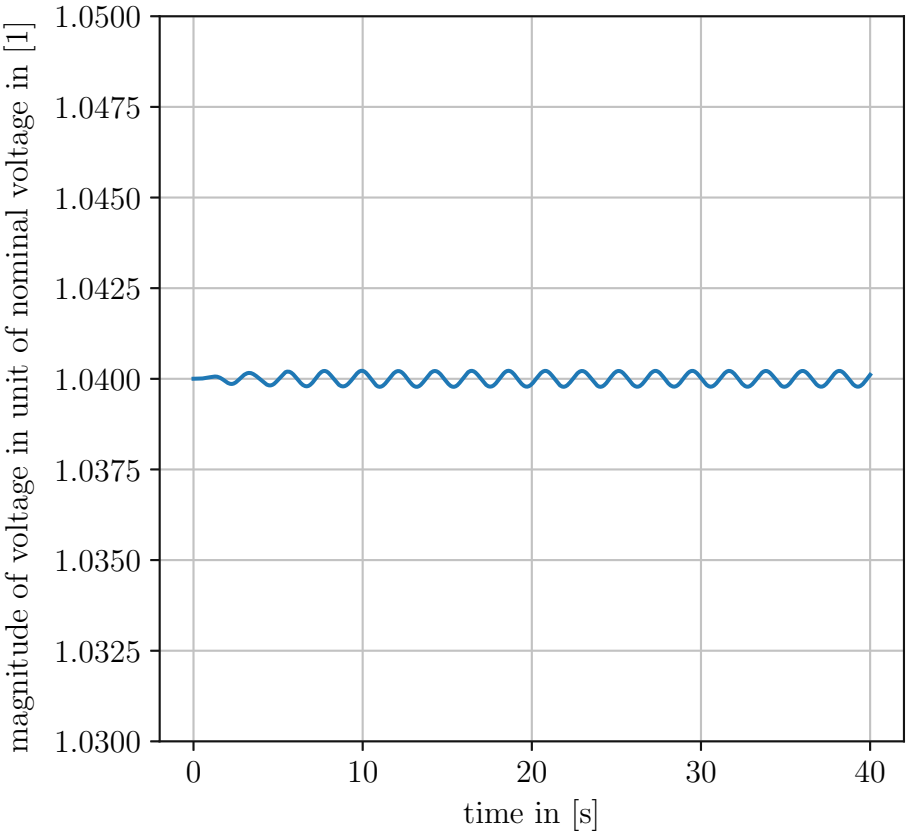

Figure 2.2: Magnitude of voltage during an oscillation event. Data is taken from [8].

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

2.3 False data injection in power grids

False data injection is a malicious attack on a power grid. Measured values - especially those from PMUs - are manipulated. The aim is to cause interventions in the power grid that endanger the stability of the power grid in order to cause power grid failures [9].

FDI is chosen as a test scenario for this thesis because it is relatively easy to create false values in an existing dataset and label them so that a machinelearning problem can be formulated with them. In the present scenario, PMU data is manipulated.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Chapter 3

Graph neural networks

Artificial neural networks, in particular convolutional neural networks - i.e., networks based on convolution operations - have made tremendous progress in recent years in various application areas. CNNs require that the underlying data has been sampled in a Euclidean space, for example pixels in an image.

This limitation can be overcome by graph neural networks. By using the edges of a graph, any irregular space can be modeled - even without knowledge of the geometry of the space. For this approach to model data, GSP offers a mature toolbox, which contains a generalization of the convolution operation towards graphs. With this motivation, in recent years GCNs were introduced to combine the advantages of CNNs and GSP.

3.1 Applications

The following section offers some examples of applications of graph neural networks in engineering and science.

An application very often cited in literature is **semi-supervised node classification**. Based on mutually shared authors (encoded in the graph topology) and used terms (encoded as graph signal), and the known subjects (encoded in the labels that are partly removed), publications are to be assigned to a subject. In the training phase random selection of labels is removed. To evaluate the model performance, another random selection of labels is masked. A commonly used setup for benchmarking in this context is a coauthered network derived from a dataset with several hundred scientific publications, such as Citeseer, Cora, Pubmed [10]. The dataset assigns each publication to exactly one discipline (in one case there are seven different disciplines) and the content and authors of the publications are assumed to be known. For example in [11] a graph is generated in such a way that each publication corresponds to a node in the graph and all publications that have

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

at least one mutually shared author are connected by an edge. Thus, there are no edges between publications without mutually shared authors. [11] generates graph signals from the content of the publications: First, all words of the general language are removed from the publications. What remains is a set of several thousand different words, each of which is typical for one or multiple subject areas. The occurrence of the words in each publication is mapped using one-hot encoding. If a word occurs at least once in a publication, the corresponding value in the graph signal is set to 1, otherwise the value is 0. The corresponding discipline is used as the label in this machine learning problem. A large part of the labels (for example 80%) is removed and the goal is to predict from the remaining labels these missing labels.

Recommender systems work in a very similar way. These can be used, for example, in social networks, online shopping or media streaming sites. In these systems, each user is represented by a node in the graph and based on the similarity of the users' behavior or their common interests, a graph is generated. Recommendations for each individual user, as to which products he could buy or movies he could watch, are then made based on the behavior of other users [12] [13] [14].

Graph neural networks also have very interesting applications in the context of **finding solutions to partial differential equations**: When applying deep learning methods to solving partial differential equations, a key challenge is to model the underlying physics. For example, when dealing with molecules and molecular potentials, these can be modeled in a natural way as graphs [15]. These methods can also be used for **drug design** [16].

In [17] it is shown how complex 3D granular flow processes in hoppers, rotating drums and mixers can be modeled by using graph neural networks. A significant challenge is to represent the complex geometry of the boundary conditions. In engineering, it is common to represent such surfaces by using triangulation. Graphs offer a way to take over this representation as boundary conditions.

Relatively little can be found in the literature on **anomaly detection** using graph neural networks. An exception is [18]. This publication proposes the so-called Graph Deviation Network. The network processes multivariate time series data. On the one hand, this network learns the structure, i.e. the edges, of the graph during training itself. On the other hand, it learns an attention-based forecast of the time series. An anomaly score is calculated from the deviation between prediction and input.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

3.2 Graph signal processing

Graph signal processing offers a toolbox for operations on graphs and graph signals. This toolbox is the basis for all kinds of graph neural networks. Therefore a quick introduction into some relevant topics of graph signal processing is given here. The contents of this section are all based on [12], [13] and [19].

3.2.1 Graphs and graph signals

[19] defines a weighted, undirected and connected graph as $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{A})$. $\mathcal{V}$ with $|\mathcal{V}| = N$ represents the set of nodes (also called vertices) of the graph, $\mathcal{E}$ the set edges and $\mathbf{A} \in \mathbb{R}_+^{N \times N}$ is the adjacency matrix. If two nodes are connected by an edge $e = \{i, j\}$, then $A_{ij} = A_{ji} = a > 0$, for unconnected nodes $A_{ij} = A_{ji} = 0$.

A graph where $A_{ij} \in \{0, 1\}$ is called unweighted graph. In case $A_{ij} \neq A_{ji}$, then the graph is directed [19].

In order to discuss graph signal processing, it is necessary to define signals over graphs. [19] introduces vectorial signals associated with graph nodes by $\mathbf{x} : \mathcal{V} \rightarrow \mathbb{R}^M$ with $\mathbf{X} \in \mathbb{R}^{N \times M}$. X_{ij} is the j^{th} value of a vectorial signal at the i^{th} node.

3.2.2 Graph shift operators and the graph Laplacian

[12] offers the following example to explain, how graph signal processing can be understood as a generalization of classical signal processing: In the special case of a uniform graph, rules of classical signal processing must be attained. Classical signal processing - more specifically, digital signal processing - considers a periodic, time-discrete (non-graph) signal $x[n]$ with period length N. This signal is described by the vector $\mathbf{x} = [x_0, x_1, x_2, \ldots, x_{N-2}, x_{N-1}]^\top$. Many operations, such as the convolution, require this signal to be shifted by a shift operator. To obtain such a shifted signal $\mathbf{x}'$, the original signal $\mathbf{x}$ is multiplied by an operator $\mathbf{A}$

$$\mathbf{x}' = \mathbf{A}\mathbf{x} \tag{3.1}$$

and the required shift can be done by using

$$\mathbf{A} = \begin{pmatrix} 0 & 0 & 0 & \ldots & 0 & 0 & 1 \\ 1 & 0 & 0 & \ldots & 0 & 0 & 0 \\ 0 & 1 & 0 & \ldots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \ldots & 1 & 0 & 0 \\ 0 & 0 & 0 & \ldots & 0 & 1 & 0 \end{pmatrix} \tag{3.2}$$

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

as shift operator. This creates a shift by one node, if a shift by multiple nodes is required, $\mathbf{A}$ is applied multiple times - i.e., a power of $\mathbf{A}$ is applied [12].

This shift operation should now be understood as an operation over a graph. A periodic, time-discrete signal can be conceived as a graph signal over a directed, circular graph - i.e., a graph where each node has a directed connection to exactly one other node. Now it turns out that the previously chosen shift operator $\mathbf{A}$ also represents the adjacency matrix for exactly this graph. This example illustrates that one possible choice for a shift operator in graph signal processing is the adjacency matrix itself [12].

Another possible choice for the graph shift operator is the graph Laplacian. [19] defines the unnormalized graph Laplacian as

$$\mathbf{L} := \mathbf{D} - \mathbf{A}, \tag{3.3}$$

where the degree matrix $\mathbf{D}$ is the diagonal matrix

$$D_{ii} = \sum_j A_{ij}. \tag{3.4}$$

When $\mathbf{L}$ acts on some graph signal $\mathbf{x} \in \mathbb{R}^N$

$$(\mathbf{Lx})_i = \sum_{j \in \mathcal{N}_i} A_{ij}[x_i - x_j], \tag{3.5}$$

its most important characteristic as a difference operator can be seen. $\mathcal{N}_i$ denotes the neighborhood of a node i, i.e. all nodes j connected by an edge to the node i. With respect to this characteristic, the graph Laplacian operator behaves exactly in same way as the classical Laplacian [19].

Furthermore, [19] provides some important characteristics of the eigenvector and eigenvalue spectrum of $\mathbf{L}$: $\mathbf{L}$ is a real symmetric matrix, therefore it has a complete set of orthonormal eigenvectors $\{\mathbf{u}_l\}_{l=0,1,\dots,N-1}$. The eigenvectors are not necessarily unique, but for all further consideration they are assumed to be fixed. The eigenvalues of $\mathbf{L}$ are real and non-negative $\{\lambda_l\}_{l=0,1,\dots,N-1}$.

The multiplicity of zero as eigenvalue is equal to the number of connected components of the graph [19].

3.2.3 Graph Fourier transform

The following explanation, how to generalize the classical Fourier transform, such that it can be applied to graphs, closely follows [19]. The classical Fourier transform is given by

$$\hat{x}(\xi) := \langle x, e^{2\pi i \xi t} \rangle = \int_{\mathbb{R}} x(t) e^{-2\pi i \xi t} dt \ . \tag{3.6}$$

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

This can be interpreted as an expansion of a function x in complex exponential functions $e^{-2\pi i\xi t}$. These complex exponential functions are the eigenfunctions of the one-dimensional classical Laplace operator:

$$-\Delta(e^{2\pi i\xi t}) = -\frac{\partial^2}{\partial t^2}e^{2\pi i\xi t} = (2\pi\xi)^2 e^{2\pi i\xi t}. \tag{3.7}$$

This observation of the classical Fourier transform builds the central point for [19] in the development of the graph Fourier transform. Therefore the graph Fourier transform is given as an expansion of a graph signal $\mathbf{x}$ in terms of the eigenfunctions $\mathbf{u}_l$ of the graph Laplacian $\mathbf{L}$

$$\hat{x}(\lambda_l) := \langle \mathbf{x}, \mathbf{u}_l \rangle = \sum_{i=1}^{N} x(i) u_l^*(i), \tag{3.8}$$

where $\hat{x}(\lambda_l)$ denotes the Fourier transformed graph signal associated with the eigenvalue λ_l. Hence, [19] gives the inverse graph Fourier transform by

$$x(i) = \sum_{l=0}^{N-1} \hat{x}(\lambda_l) u_l(i). \tag{3.9}$$

In the classical Fourier transform, ξ (or $2\pi\xi$) represents a frequency. In analogy, the eigenvalues λ_l and eigenvectors $\mathbf{u}_l$ of the graph Laplacian $\mathbf{L}$ are also associated with a generalized frequency: Eigenvectors belonging to eigenvalues close to 0 change only slowly over the whole graph. That is, within the neighborhood $\mathcal{N}_i$ of a node i, the value of the graph signal represented by $\mathbf{u}_l$changes little. The eigenvector $\mathbf{u}_0$ to the eigenvalue $\lambda = 0$ is constant throughout the whole graph and has a value of $1/\sqrt{N}$. Eigenvectors belonging to eigenvalues far from 0 change very much within the neighborhood of a node and have many oscillations over the whole graph [19].

3.2.4 Graph convolution

Graph convolution is defined by [12] by the matrix vector multiplication

$$\mathbf{y} = \mathbf{G}\mathbf{x} = g(\mathbf{A})\mathbf{x} = \sum_{k=0}^{K} \theta_k \mathbf{A}^k \mathbf{x}, \quad K < N, \tag{3.10}$$

with the graph signal $\mathbf{x}$, $g(\mathbf{A})$ a polynomial filter (i.e. shift operator) of degree K and filter coefficients θ_k. Graph convolution results in k^{th} iteration propagating the signal from all k-hop neighbors to a node - weighted by a filter coefficient θ_k. This

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

behavior is analogous to classical convolution - reduced by all that a graph lacks in structure such as direction and distance. [12] states, that, for practical reasons, $\mathbf{A}$ is often normalized by division by $|\lambda_{max}|$, where λ_{max} is the eigenvalue of $\mathbf{A}$ with the greatest magnitude. Alternatively, normalization can also be performed by

$$\bar{\mathbf{A}} = \mathbf{D}^{-(1/2)}\mathbf{A}\mathbf{D}^{-(1/2)}, \tag{3.11}$$

where $\mathbf{D}$ is the degree matrix. [12] reasons this with an increase of computational stability of $\mathbf{G}$ because all (non-maximum) eigenvalues are inside the unit circle.

Now the convolution of Fourier-transformed signals are considered. Therefore an eigendecomposition is applied to the shift operator

$$\mathbf{A} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^\top, \tag{3.12}$$

with the orthogonal eigenvector matrix $\mathbf{U} \in \mathbb{R}^{NxN}$ and the diagonal eigenvalue matrix $\mathbf{\Lambda} \in \mathbb{R}^{NxF}$ ordered by $\lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_{N_1}$. The graph signal is transformed to the frequency domain by

$$\hat{\mathbf{x}} = \mathbf{U}^\top\mathbf{x}. \tag{3.13}$$

As shown by [13], the Fourier transformed output of the graph convolution yields

$$\begin{aligned}\hat{\mathbf{y}} =& \mathbf{U}^\top\mathbf{y} \\ =& \sum_{k=0}^{K} \theta_k \mathbf{U}^\top\mathbf{U}\mathbf{\Lambda}^k\mathbf{U}^\top\mathbf{x} \\ =& \sum_{k=0}^{K} \theta_k \mathbf{\Lambda}^k\hat{\mathbf{x}} \\ =& \mathbf{G}_\theta(\mathbf{\Lambda})\hat{\mathbf{x}}.\end{aligned} \tag{3.14}$$

$\mathbf{G}_\theta(\mathbf{\Lambda})$ is a diagonal matrix $\mathrm{diag}(\theta(\lambda_i))$, with

$$\theta(\lambda) = \sum_{k=0}^{K} \theta_k \lambda^k. \tag{3.15}$$

3.3 Types of graph neural networks

Basically, it is possible for graph neural networks to give output at the level of nodes, edges, or the whole graph. Another fundamental classification is that graph neural networks either process signals over the graph or the structure of the graph itself [14].

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

For further consideration, the focus is on those graph neural networks that make outputs at the node level and process graph signals.

The vast majority of graph neural networks that are of practical importance recursively aggregate feature vectors of neighboring nodes (message passing). After k iterations, the transformed graph signal of a given node i is an aggregation of the feature vectors of the k-hop neighbors $\mathcal{N}_i$ - with respect to the structural properties of the graph [20].

Explaining a complete taxonomy of all graph neural networks is a complicated matter. At this point, only a rough overview is given. A comprehensive taxonomy can be found in [21].

- **Recurrent graph neural networks:**

  According to [21], recurrent graph neural networks are those graph neural networks that were developed earliest. These types of networks learn node representations through recurrent neural architectures. The idea behind this is that information is exchanged until a stable equilibrium is reached between the nodes. Today, recurrent graph neural networks are important mainly because they have conceptually influenced many other developments in the field. The idea of message-passing originates from these networks.

  [21] provides as examples for recurrent graph neural networks [22], [23], [24] and [25].

- **Graph convolutional neural networks:**

  Graph convolutional neural networks use the generalization of the convolution operation over graphs. There are networks that perform convolution in the node domain as well as those that perform convolution in spectral space. [14] [13] The following Section 3.4 explains this type of network in more detail.

  Among many others, [26], [11], [27], [28], [29], [30] and [31] are examples of this type of networks listed by [21].

- **Graph autoencoders:**

  Graph autoencoders follow the example of classic autoencoders. It is an unsupervised learning method. A representation of nodes or the whole graph is learned in a latent vector space. The original graph or graph signals are then reconstructed from this representation. These networks are used for example in network embedding or graph generation.

  Examples of graph autoencoders for network embedding provided by [21] are [32], [33], [34], [35], [36] and [37]. According to [21], examples for graph generation are [38], [39], [40], [41] and [42].

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

- **Spatial-temporal graph neural networks:**

  Spatial-temporal graph neural networks process graph signals that are time-dependent. The time domain and the spatial (i.e. vertex) domain are combined. Such an architecture is suitable, for example, for modeling traffic flows or water levels in connected water systems. Often recurrent neural networks or convolutional neural networks are used to represent the time dependencies.

  [21] provides as examples of spatial-temporal graph neural networks [43], [44], [45], [46], [47] and [48].

A comprehensive survey that includes benchmark tests for various graph neural network can be found in [49].

3.4 Graph convolutional neural networks

The experimental part of this thesis deals with a specific implementation of a graph convolutional neural network. For this reason, a more detailed introduction to this type of networks is given here.

3.4.1 From Convolutional Neural Networks to Graph Convolutional Neural Networks

Graph convolutional neural networks take advantage of the fact that there is a generalization of the convolution operation to graphs through the tools of graph signal processing. This allows the established concepts of convolutional neural networks, which have been so successful, to be used to process data sampled in a non-Euclidean domain.

Just like classical convolutional neural networks, the idea that connects all the different convolutional neural networks is that the network learns certain filters.

In the case of classical convolutional neural networks, the input signal is convolved with small, spatially bounded filters. This allows efficient searching of a large amount of input data for specific patterns. The result of the convolution is transformed by a non-linear function. Examples of this non-linear function are the sigmoid function

$$\mathrm{sig}(x) = \frac{1}{1 + e^{-x}}, \tag{3.16}$$

or the ReLU function

$$\mathrm{ReLU}(x) = \max(0, x). \tag{3.17}$$

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

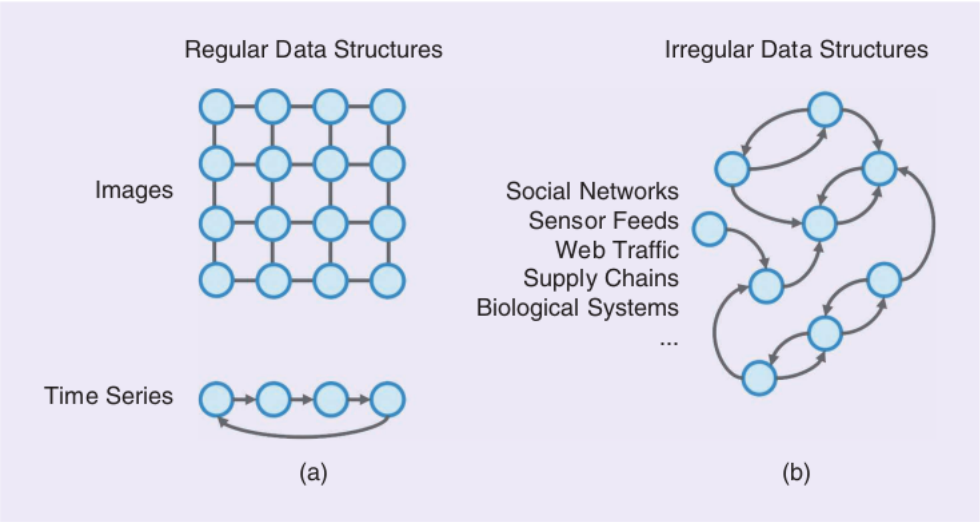

Figure 3.1: (a) Shows examples for regular data structures, namely a 2D image and a 1D, periodic time series. (b) Shows an example of data represented as a graph. Examples for such data can be social networks, sensor feeds, web traffic, supply chains or biological systems. This figure is taken from [12].

The combination of convolution and non-linear transformation forms a CNN unit. After using a CNN unit, the dimension of the input signal is often reduced by pooling. In practice, deep CNNs are often used, where several CNN and pooling layers are executed in succession. The output signal of the previous layer is used as input signal for the next layer. Fig. 3.2 shows an example of a classical CNN architecture [14].

However, a graph provides much less structure than does an Euclidean space. For example, when processing images with a classical CNN, it is clear how neighboring pixels are oriented to each other, one is on the left, one on the right, one above, one below. Thus there are clearly defined directions. This is not true for graphs. Fig. 3.1 gives examples for regular, Euclidean data structures and irregular graph structures.

Likewise, distance measures exist only to a very limited extent. Within the immediate neighborhood of a node $\mathcal{N}_i$, the weight of an edge can be interpreted as an inverse distance. Outside the immediate neighborhood of a node, distances are even more difficult to determine. These two aspects are examples of the new problems that graph convolutional neural networks have to master. In the following, these topics will be discussed.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

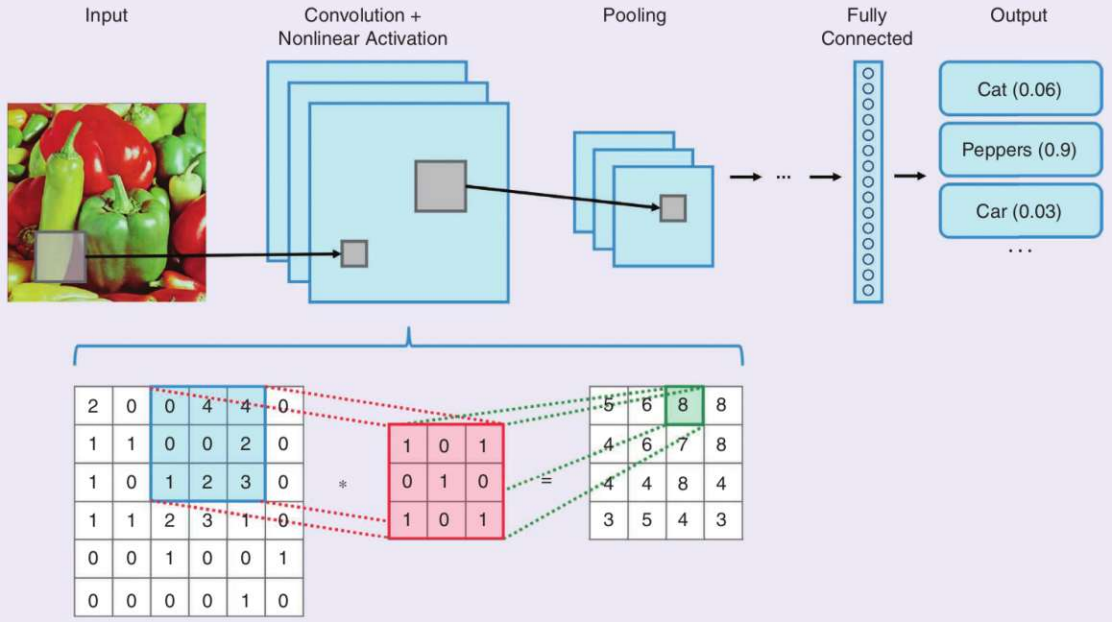

Figure 3.2: As an example of a classical CNN architecture a classification setup to distinguish photographs of cats, peppers and cars is shown. Steps of a classical CNN can be seen from left to right. First the input signal is convolved with multiple different filters and the result transformed by a non-linear activation function. Then pooling is executed to reduce the dimensionality of the problem. For practical reasons, after multiple convolution and pooling layers some fully connected layers are used to transform the result into a usable format. The image is taken from [12].

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

3.4.2 Spectral domain methods

There are basically two ways to implement a graph convolutional neural network. Either the convolution can be performed in the node domain or in the spectral domain.

In the spectral variant, the input signal $\mathbf{x}$ is first represented in the spectral domain as $\hat{\mathbf{x}}$ by application of the graph Fourier transformation Eq. (3.8). In the spectral domain, the convolution operation is performed as a multiplication with a trainable filter. Afterwards the back-transformation into the node domain is carried out with the inverse graph Fourier transform Eq. (3.9).

The costly eigendecomposition of the graph Laplacian is a big problem of this variant. For practical applications, approximations must be used for this operation. For example, [26] uses Chebyshev polynomials to approximate the graph Fourier transform [12] [14].

In Section 3.5, the implementation of a graph convolutional neural network that performs the approximation of a spectral convolution is discussed.

3.4.3 Node domain methods

Alternatively graph convolutional neural networks can perform the convolution operation directly in the node domain. Therefore Eq. (3.10) is used. [12] states

$$\mathbf{x}' = \sigma(g(\mathbf{A})\mathbf{x}) \tag{3.18}$$

as the general form of all graph convolutional neural layers operating with $\mathbf{x}$ as input signal, $\mathbf{x}'$ as the output signal, $\sigma(\cdot)$ a non-linear activation function and $g(\mathbf{A})$ a trainable, polynomial filter.

3.4.4 Graph pooling and graph aggregation

Pooling is a form of down-sampling, it reduces the number of nodes in a graph. There are two main reasons why it is useful to perform such an operation: Reducing the dimensionality of the problem and hierarchical learning. [12] provides several examples to conduct graph pooling like Sort Pooling [50], Differentiable Pooling [51], Top-k Pooling [52] or Self-Attention Graph Pooling [53].

Classical CNNs often end up with a sequence of several dense layers, for example when an image is to be classified as a whole. In graph convolutional neural networks, such a step cannot be performed. The underlying graphs may be of different sizes and the nodes may be permuted. A resulting vector would be arbitrary and could not be used as input to a dense layer. Therefore, there is the operation of graph aggregation, often referred to as the final graph pooling layer. A mean or

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

sum operation is often used for this purpose. Fig. 3.3 shows the difference between graph pooling and graph aggregation [12].

3.5 GCN framework

Since its publication in 2016, Thomas Kipf and Max Welling's GCN (Graph Convolution Network) [11] has become probably the most widely used implementation of a graph neural network. A modification of GCN is used in the experimental part of this thesis. Therefore, the derivation of the used approximation of the convolution operation and the design of the graph convolutional layer is now explained in detail.

[11] defines a layer-wise propagation rule for the multilayer network by

$$\mathbf{H}^{(l+1)} = \sigma(\tilde{\mathbf{D}}^{-1/2}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-1/2}\mathbf{H}^{(l)}\mathbf{W}^{(l)}). \tag{3.19}$$

$\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}_N$ is the adjacency matrix of an undirected graph with added self connections, $\tilde{D}_{ii} = \sum_j \tilde{A}_{ij}$ in analogy to Eq. (3.4) and $\mathbf{W}^{(l)}$ is a layer-specific trainable weight matrix. $\sigma(\cdot)$ is the non-linear activation function like Eq. (3.16) or Eq. (3.17). $\mathbf{H}^{(l)} \in \mathbb{R}^{N\times D}$ is the matrix of activations (i.e. inputs) in the l^{th} layer, with N the number of nodes and D the number of input channels. $\mathbf{H}^{(0)} = \mathbf{X}$ is the input signal into the first layer.

As a next step, [11] designs a convolution layer. First, convolution in the spectral domain Eq. (3.14) is rewritten as

$$\mathbf{g}_\theta * \mathbf{x} = \mathbf{U}\text{diag}(\hat{\theta})\mathbf{U}^\top\mathbf{x} = \mathbf{U}\mathbf{G}_\theta(\mathbf{\Lambda})\mathbf{U}^T\mathbf{x}, \tag{3.20}$$

where $\mathbf{g}_\theta = \text{diag}(\theta)$ is a filter parametrized by $\theta \in \mathbb{R}^N$ in the spectral domain. $\mathbf{U}$ is the matrix of eigenvectors of the unnormalized graph Laplacian and $\mathbf{G}_\theta(\mathbf{\Lambda})$ is a matrix of trainable parameters as given in Eq. (3.15). The unnormalized Laplacian $\mathbf{L}$ Eq. (3.3) is replaced by the normalized graph Laplacian defined by [11] as

$$\bar{\mathbf{L}} = \mathbf{I}_N - \bar{\mathbf{A}} = \mathbf{I}_N - \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}. \tag{3.21}$$

In order to avoid a too complicated notation, the eigensystem of $\bar{\mathbf{L}}$ is still denoted by λ and $\mathbf{u}$ without bars.

The repeated calculation - as it is required during the training of the graph neural network - of the convolution as defined in Eq. (3.20) is computationally very expensive. The complexity of the matrix multiplication with $\mathbf{U}$ is $\mathcal{O}(N^2)$ and the eigendecomposition is even more complex. In order to find a computationally more efficient solution, [11] approximates $\mathbf{g}_\theta(\mathbf{\Lambda})$ - based on [54] - by a truncated expansion in terms of the Chebyshev polynomials $\mathbf{T}_k(x)$ up to order K

$$\mathbf{g}'_\theta(\Lambda) = \sum_{k=0}^{K} \theta'_k \mathbf{T}_k(\tilde{\mathbf{\Lambda}}). \tag{3.22}$$

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

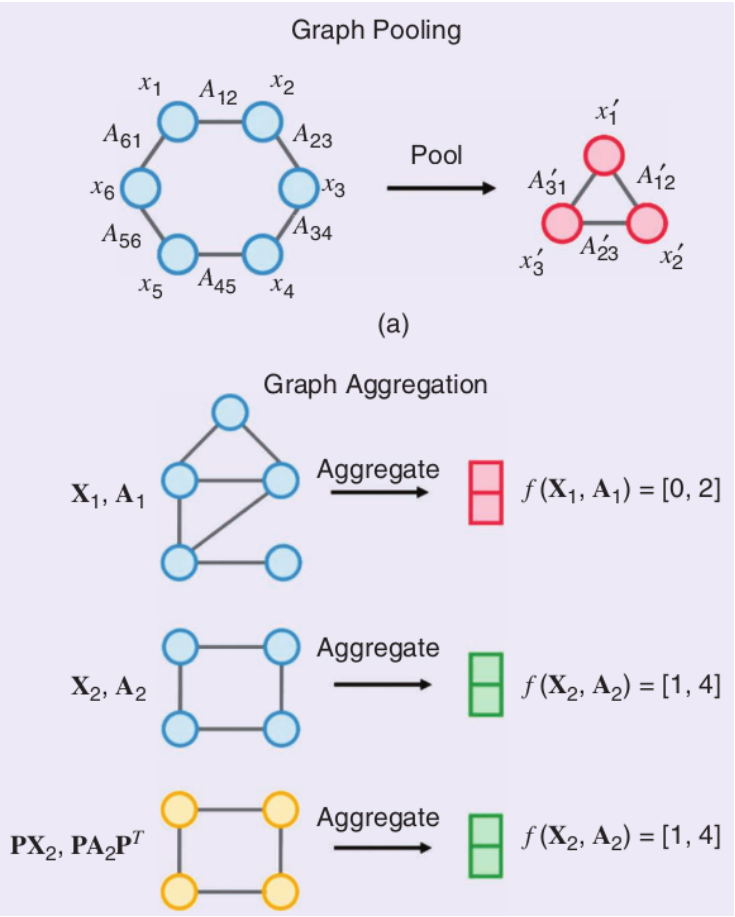

Figure 3.3: Graph pooling and graph aggregation. (a) Graph pooling accepts a graph signal and produces a new, representative graph signal indexed by a smaller graph support. (b) Global aggregation can accept graphs of potentially varying sizes and produce fixed-length, representative vectors. This figure is taken from [12].

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

[11] rescales $\mathbf{\Lambda}$ to

$$\tilde{\mathbf{\Lambda}} = \frac{2}{\lambda_{\max}}\mathbf{\Lambda} - \mathbf{I}_N. \tag{3.23}$$

This step is reasoned with an improved numerical stability. θ'_k is now a vector of Chebyshev polynomials. The Chebyshev polynomials are recursively defined as

$$\begin{aligned} T_k(x) &= 2xT_{k-1}(x) - T_{k-2}(x), \\ T_0(x) &= 1, \\ T_1(x) &= x. \end{aligned} \tag{3.24}$$

As shown by [11], an approximation of Eq. (3.20) is now given by

$$\mathbf{g}_\theta * \mathbf{x} = \mathbf{g}'_\theta(\Lambda) = \sum_{k=0}^{K} \theta'_k \mathbf{T}_k(\tilde{\mathbf{L}})x, \tag{3.25}$$

with $\tilde{\mathbf{L}} = \frac{2}{\lambda_{\max}}\bar{\mathbf{L}} - \mathbf{I}_N$. Here $(\mathbf{U}\mathbf{\Lambda}\mathbf{U}^\top)^k = \mathbf{U}\mathbf{\Lambda}^k\mathbf{U}^\top$ is used. The approximation of a convolution in Eq. (3.25) is now K-localized because it is a K^{th} order polynomial in the normalized Laplacian $\tilde{\mathbf{L}}$, i.e. only information from a K-hop neighborhood propagates to a node i. This approximation reduced the complexity of the problem to $\mathcal{O}(N)$. The approximation up to this point took [11] from [26].

In order to reduce the number of trainable parameters and to make the behavior of this graph convolution layer more localized, [11] chose $K = 1$. Because of $T_1(x) = x$ the operation becomes linear in $\tilde{\mathbf{L}}$. To further reduce the complexity of the calculation $\lambda_{\max} \approx 2$ is used. [11] justifies this step by the argument that the parameters θ can adapt to this choice. This leads to the new expression

$$\begin{aligned} \mathbf{g}_\theta * \mathbf{x} &\approx \theta'_0\mathbf{x} + \theta'_1(\tilde{\mathbf{L}} - \mathbf{I}_N)\mathbf{x}, \\ &\approx \theta'_0\mathbf{x} + \theta'_1\mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}\mathbf{x}. \end{aligned} \tag{3.26}$$

The trainable parameters are θ'_0 and θ'_1. As for a classical CNN, the trainable parameters are shared across the whole graph. By the successive application of l layers of this kind, information propagates from a l-hop neighborhood to a node i.

[11] reduces the number of parameters one step further Eq. (3.26) to

$$\mathbf{g}_\theta * \mathbf{x} \approx \theta(\mathbf{I}_N + \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2})\mathbf{x}, \tag{3.27}$$

with only one trainable parameter $\theta = \theta'_0 = -\theta'_1$. All eigenvalues of $(\mathbf{I}_N + \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2})$ are within the $[0, 2]$. [11] explains, that repeated application of Eq. (3.27) in a deep neural network can lead to numerical instabilities like exploding or vanishing gradients. To handle this problem the renormalization $(\mathbf{I}_N + \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}) \to \tilde{\mathbf{D}}^{-1/2}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-1/2}$ with $\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}_N$ and $\tilde{\mathbf{D}} = \sum_j \tilde{\mathbf{A}}_{ij}$ is applied by [11].

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

Now the input signal is generalized from $\mathbf{x} \in \mathbb{R}^N$ to $\mathbf{X} \in \mathbb{R}^{N \times C}$ with the number of input channels C. Moreover F different filters or feature maps per layer can be used with

$$\mathbf{Z} = \tilde{\mathbf{D}}^{-1/2}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-1/2}\mathbf{X}\mathbf{\Theta}, \tag{3.28}$$

with the filter parameter matrix $\mathbf{\Theta} \in \mathbb{R}^{C \times F}$ and the convolved signal $\mathbf{Z} \in \mathbb{R}^{N \times F}$. Note that $\mathbf{\Theta}$ combines signals from different input channels and different nodes. The computational complexity of this operation is now $\mathcal{O}(NFC)$, as $\tilde{\mathbf{A}}\mathbf{X}$ can be implemented efficiently as a product of a sparse matrix with a dense matrix.

The full propagation rule for a GCN from layer l to layer $(l+1)$ is given by [11] as

$$\mathbf{H}^{(l+1)} = \sigma(\tilde{\mathbf{D}}^{-1/2}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-1/2}\mathbf{H}^{(l)}\mathbf{\Theta} + \mathbf{B}), \tag{3.29}$$

where $\mathbf{B} \in \mathbb{R}^{N \times F}$

$$\mathbf{B} = \begin{pmatrix} \mathbf{b} \\ \mathbf{b} \\ \vdots \\ \mathbf{b} \end{pmatrix}, \tag{3.30}$$

with $\mathbf{b}^\top \in \mathbb{R}^F$ is the vector of biases. $\mathbf{H}^{(l+1)} = \sigma(\dots)$ is a shorthand notation for

$$\mathbf{H}^{(l+1)} = \begin{pmatrix} \sigma(\dots) & \sigma(\dots) & \dots & \sigma(\dots) \\ \sigma(\dots) & \sigma(\dots) & \dots & \sigma(\dots) \\ \vdots & \vdots & \ddots & \vdots \\ \sigma(\dots) & \sigma(\dots) & \dots & \sigma(\dots) \end{pmatrix}. \tag{3.31}$$

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

Chapter 4

GCN for anomaly detection

The aim of this thesis is to detect anomalies in data sampled in power grids. The chosen formulation as a machine learning problem (cf. Chapter 5) requires GCN to be able to detect rapid changes in the signal in the node domain. This chapter will address the resulting problem.

4.1 Deficiencies of GCN

As explained in Chapter 5, the anomalies in the phasor measurement unit data that are to be detected are point anomalies. These anomalies are to be detected by means of a classification architecture. Therefore, it is necessary that GCN can handle rapid changes of the signal in the node domain.

Now the fact is that GCN was developed for semi-supervised node classification. In such a setup, it is rather desirable to compensate differences between data at neighboring nodes. Accurate detection of differences is not required. Semi-supervised node classification requires a network that has some low-pass characteristic in the node domain, but for (point) anomaly detection this low-pass characteristic is detrimental. If the signal at a single node is an outlier, the outlier signal would be smeared over the neighboring nodes and the outlier could not be detected.

The simplification that is done from Eq. (3.26) to Eq. (3.27) leads to the behavior, that is problematic for detecting point outliers in data. In Eq. (3.27) $(\mathbf{I}_N + \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2})\mathbf{x}$ creates a weighted average of the signal at a center node $(\mathbf{I}_N)$ and all its neighboring nodes $(\mathbf{A})$. The artificial neural network has no chance to learn filters in such a way, that would provide an output signal, when differences between a center node and the neighboring nodes occur.

To my knowledge, there are no publications describing how GCN or any other graph convolutional neural network can be used to detect point-wise outliers in

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

data. GCN thus has a strong tendency to over-smoothing, which has received little attention in the literature. These deficiencies are the reason for the contributions of this thesis, that are described in the following section.

4.2 Adapting GCN

The goal is to develop a graph convolution layer that is able to detect rapid changes of signals in the node domain. The basis for an adapted GCN layer is Eq. (3.26). The sign of the second term is switched so that

$$\mathbf{g}_\theta * \mathbf{x} \approx \theta'_0\mathbf{x} - \theta'_1\mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}\mathbf{x}, \tag{4.1}$$

and the vectorization in analogy to Eq. (3.28) signals leads to

$$\mathbf{Z} = \mathbf{X}\boldsymbol{\Theta}_\mathbf{0} - \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}\mathbf{X}\boldsymbol{\Theta}_\mathbf{1}. \tag{4.2}$$

If corresponding filter parameters $\Theta_{0;ij}$ and $\Theta_{1;ij}$ have different sign, then an output similar to the original GCN results is produced. That is, the output is a weighted mean of the signal at a central node and at the signal of the neighboring nodes. However, the desired behavior is as follows: If corresponding entries have equal sign, a meaningful output is produced only when there is a difference between the signal at a central node and the average neighboring signals. We observe that the proposed expression is very similar to a graph Laplacian.

In Eq. (3.19), GCN replaces $\mathbf{A}$ by $\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}_N$, with the goal of changing the structure of the eigenvalue spectrum to allow a more stable execution in a deep neural network. This change cannot be made in the adapted version because the introduction of self-loops would cancel for the differences between a central node and the neighboring nodes.

The full propagation rule for the adapted GCN yields in analogy to Eq. (3.29)

$$\mathbf{H}^{(l+1)} = \sigma(\mathbf{H}^{(l)}\boldsymbol{\Theta}_\mathbf{0} - \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}\mathbf{H}^{(l)}\boldsymbol{\Theta}_\mathbf{1} + \mathbf{B}). \tag{4.3}$$

Chapter 6 compares the two GCN variants in terms of their anomaly detection performance.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

Chapter 5

Experimental setup

The goal of this thesis is to detect point-wise anomalies in phasor measurement unit data collected in a power grid. This chapter shows how this problem is formulated as a machine learning problem to obtain a trainable system, how the parameters of the deep-learning model are selected, how the dataset is transformed into a usable data-structure, and some further preparation of the data.

5.1 Formulation as a machine learning problem

The most common neural network architecture for anomaly detection is an auto-encoder. This architecture uses the same data as input and output. In between, a deep neural network is used. Starting from the input, the dimensionality of the problem is reduced more and more by using layers with less and less trainable parameters. This results in all input information being encoded in a latent state with much lower dimensionality. Subsequent to the latent state, essentially the same sequence of layers as on the input side is used in a mirror image fashion. Only normal data is used for training. The neural network thus learns to encode normal input information into a state of lower dimensionality and to reproduce it from there. To detect anomalies, the input is compared with the output. For normal data, there is not much deviation between input and output if the training has worked well. For anomalous data, however, there are large deviations, because the neural network is not able to correctly endcode and decode anomalous data.

However, most of the literature describes applications of GCN for semi-supervised node classification - i.e., a classification problem. For this reason, a formulation as a classification problem is also chosen in this thesis. It is a binary classification problem, i.e. a certain datum is either normal or anomalous. Formulating an anomaly detection problem as a classification problem introduces a significant problem, namely that the data is typically very unbalanced - there is much, much

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

more normal data than anomalous data. However, by using appropriate class weights, this problem can be handled well. The formulation as a classification problem yields the multilayer graph convolutional neural network

$$f : \mathbb{R}^{N \times F} \to \mathbb{R}^{N \times 2}, \tag{5.1}$$

with the number of input features F and a two-dimensional output space, one dimension for normal the other for anomalous.

5.2 Model design

For the experiments performed, a model consisting of two consecutive GCN layers is used. The first GCN layer uses 4 input features (c.f. Section 5.4) and produces 16 latent features as output. The second GCN layer uses as input the 16 latent features and has a two-dimensional output - normal and anomalous. Thus, the sequence of dimensions in the layers two layer is

$$\mathbb{R}^{N \times 4} \xrightarrow{\text{GCN}_1} \mathbb{R}^{N \times 16} \xrightarrow{\text{GCN}_2} \mathbb{R}^{N \times 2}, \tag{5.2}$$

with N the number of nodes in the graph, and GCN_1 and GCN_2 the two GCN layers. A pooling or aggregation layer is not used because an output is required for each individual node.

The first GCN layer uses as a non-linear activation function the ReLU function Eq. (3.17). The second GCN layer uses a logarithmic softmax function

$$\text{LogSoftmax}(x_i) = \log\left(\frac{\exp(x_i)}{\sum_j \exp(x_j)}\right), \tag{5.3}$$

with $i, j \in \{1, 2\}$ for binary classification, as non-linear activation function in order to retrieve an output that is suitable for a classification problem.

Between the first and second GCN layer a drop-out layer is used. During training data samples are zeroed out with a probability of $p = 0.5$. This is an effective technique of regularization and helps to prevent co-adaption of neurons [55].

The used loss function is a negative log likelihood function with mean reduction

$$\ell(x, y) = \sum_{n=1}^{N} \frac{-w_{y_n} x_{n,y_n}}{\sum_{n=1}^{N} w_{y_n}}, \tag{5.4}$$

with $\mathbf{x} \in \mathbb{R}^{N \times 2}$ the output of the log Softmax function, $\mathbf{y} \in \mathbb{R}^{N \times 2}$ and $y_{ij} \in [0, 1]$ the one-hot encoded targets and $\mathbf{w} \in \mathbb{R}^2$ the class weights. Class weights are

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

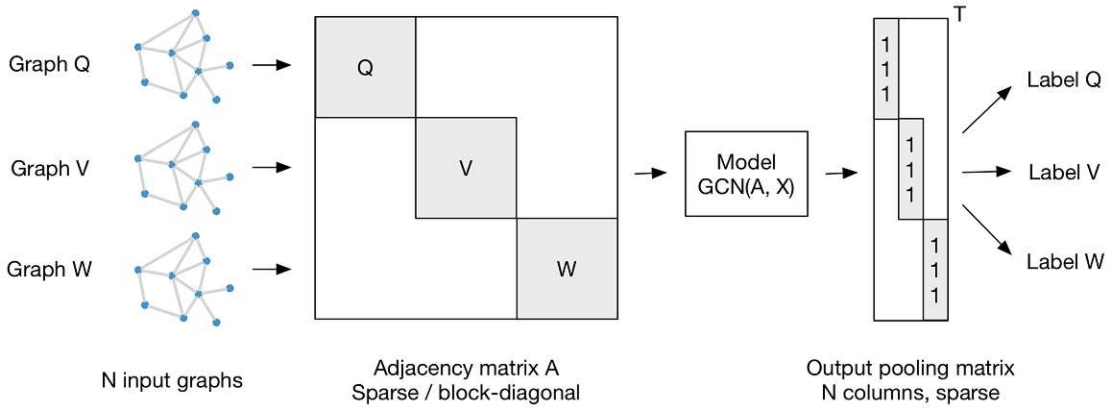

Figure 5.1: The adjacency matrix is repeated several times in a block-diagonal matrix to enable batch-wise calculation for weights and biases. This figure is taken from [11].

defined as the share of samples of one class in the total samples

$$w_i = \frac{N_s^{(i)}}{N_s}, \tag{5.5}$$

with w_i the weight of class i, $N_s^{(i)}$ the number of samples in class i and N_s the total number of samples.

The initialization of weights is done with a Glorot initializer (see [56]), biases are initialized to zero and the optimization is carried out by using an Adam optimizer (see [57]).

In order to update weights and biases based on gradients calculated not only for one data sample, i. e. one value for every node and channel and one label for every node, but to have multiple samples, data samples are stacked into batches. Therefore [11] proposes to repeat the adjacency matrix in a block diagonal matrix as well as graph signals and labels in vectors multiple times. Due to the efficient usage of sparse matrices, a batch size of $n = 32$ can be used. Fig. 5.1 explains details of this step.

The experiments are performed using PyTorch Geometric (see [2]). This is an open source framework for geometric machine learning written in Python. Thanks to the fact that Pytorch Geometric is very well documented and programmed in an understandable way, it is feasible to implement various experimental changes to the GCN layer.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

5.3 Dataset

The base dataset to perform experiments on is taken from [8] - a test cases library for methods locating the sources of sustained oscillations in a power grid. This library contains several example dataset, all built over a power grid depicted in Fig. 5.2. This grid with 179 busses is a simplified representation of a power grid in north-eastern United States. The data was generated via simulations in [8] and provides 40 seconds of a poorly damped failure event, where severe oscillations in voltage and current occur.

However, in terms of false data injection, the data describing an oscillation event are considered to be normal, data. A physically possible state of the system is represented. As will be described in the following section, additional false data injection anomalies are generated in this data.

Thus, a dataset that already represents a faulty operating condition of a power system is searched for additional inserted faults. One could now argue that the reason for this procedure is that it would be too easy to search for false data anomalies in a completely normal state, that the neural network should be tested under tough conditions, or that the correct detection of a false data injection attack within an existing error case would be especially critical. However, these considerations are not the real reason why this dataset was manipulated with further false-data injection anomalies. Rather, the real reason is that it was not possible to find data of a power grid in a normal operating state. Therefore, it was necessary to use on exactly this data set.

The underlying nominal grid frequency in the data set is 60 Hz because it is an American grid. The data is sampled with 30 Hz and one dataset contains 1200 time steps, i.e. 40 seconds. The datasets provides amplitude and angle of voltage for every of the 179 busses and the rotor speed is only provided for some busses and not used as an input feature. Therefore, the rotor speed data is removed, because the neural network requires that input channels are available for all nodes. Moreover, magnitude and angle of current are provided for all transmission lines. However, this current data is not used for experiments because this data would represent graph signals associated with edges, not nodes. The chosen graph architecture uses graph signals only for nodes, i.e. voltage signals from busses. As an example, Fig. 2.1 and Fig. 2.2 show all 40 seconds of voltage signals for one bus.

5.4 Data preparation

The grid structure, as it is shown in Fig. 5.2, is encoded into a graph adjacency matrix $\mathbf{A}$. Every bus represents a node, every transmission line represents an edge. For all edges a weight of $a_{ij} = 1$ is chosen, so an unweighted graph is implemented.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

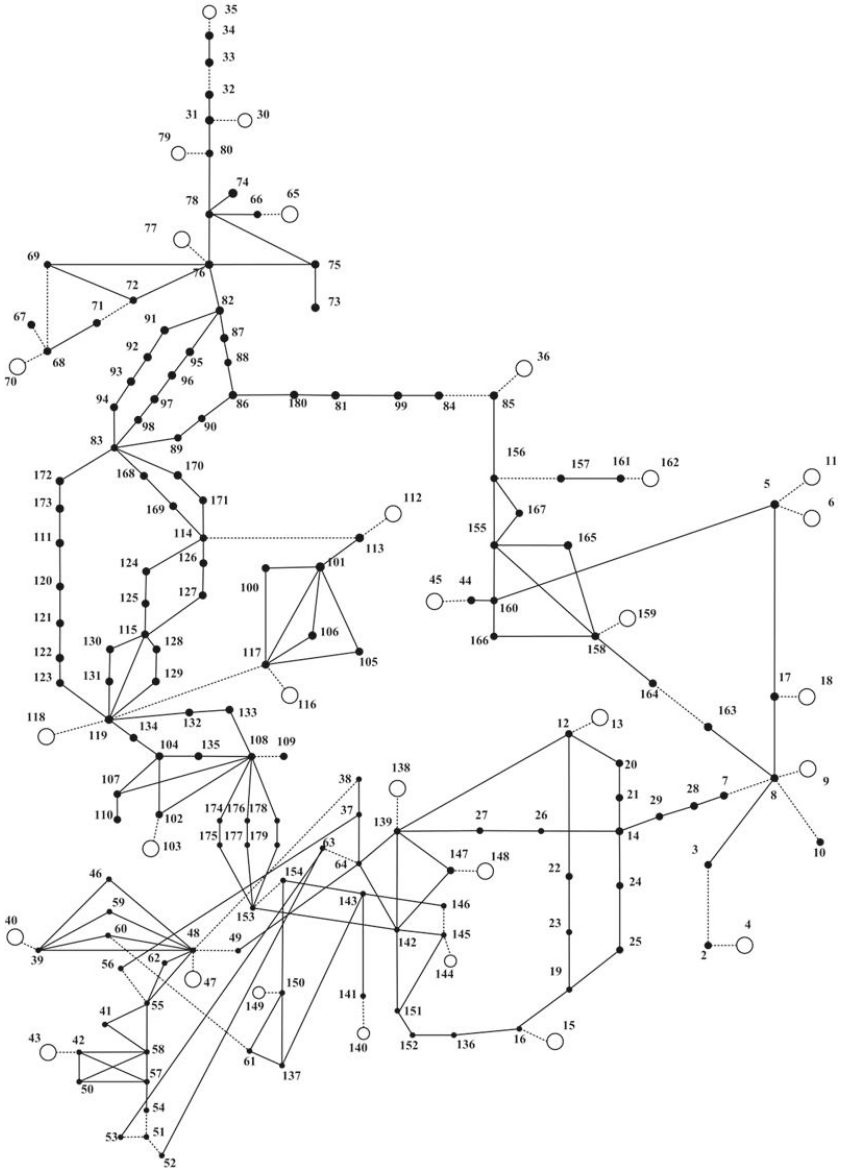

Figure 5.2: The WECC-179 bus system is a simplified representation of the power grid in north-eastern United States. All experiments in this thesis are performed with simulated data from this grid. This figure is taken from [8].

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

The resulting adjacency matrix is visualized in Fig. 5.3.

As explained in the previous section only voltage magnitude and angle are used. Current values are discarded because they are associated with edges instead of nodes. Rotor speed values are not used because they exist only for a subset of nodes. The resulting shape of the graph signal matrix is $\mathbf{X} \in \mathbb{R}^{2\times 179}$. The resulting graph signals for a fixed point in time are shown in Fig. 5.4 and Fig. 5.5. Each of the 1200 time steps is used as an independent sample $\mathbf{X}_i$. This means that only the relationship between the nodes for one time step is considered, but not the time course of the signals. All 1200 independent samples $\mathbf{X}_i$ form the input dataset $\{\mathbf{X}_1, \mathbf{X}_2, ..., \mathbf{X}_{1200}\}$.

Data is disturbed by point-wise anomalies. With a probability of $p = 0.05$ the angle of one node is increased by 20°

$$\tilde{x}_{0j} = x_{0j}, \qquad \text{with } p = 0.95, \tag{5.6}$$
$$\tilde{x}_{0j} = x_{0j} + 20°, \qquad \text{with } p = 0.05, \tag{5.7}$$

with the voltage angle x_{0j} of a node j. independently, the voltage magnitude of a node is increased by 0.1 (in units of the nominal voltage magnitude)

$$\tilde{x}_{0j} = x_{1j}, \qquad \text{with } p = 0.95, \tag{5.8}$$
$$\tilde{x}_{0j} = x_{1j} + 0.1, \qquad \text{with } p = 0.05, \tag{5.9}$$

with the voltage magnitude x_{1j} of a node j.

Before training the data is normalized separately for each of the two input channels such that the mean of the normalized signal $\text{mean}(\mathbf{x}'_\mathbf{i}) = 0$ and the standard deviation $\text{std}(\mathbf{x}'_\mathbf{i}) = 1$.

No train test split is done before training, so the same data is used for training and testing the model.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

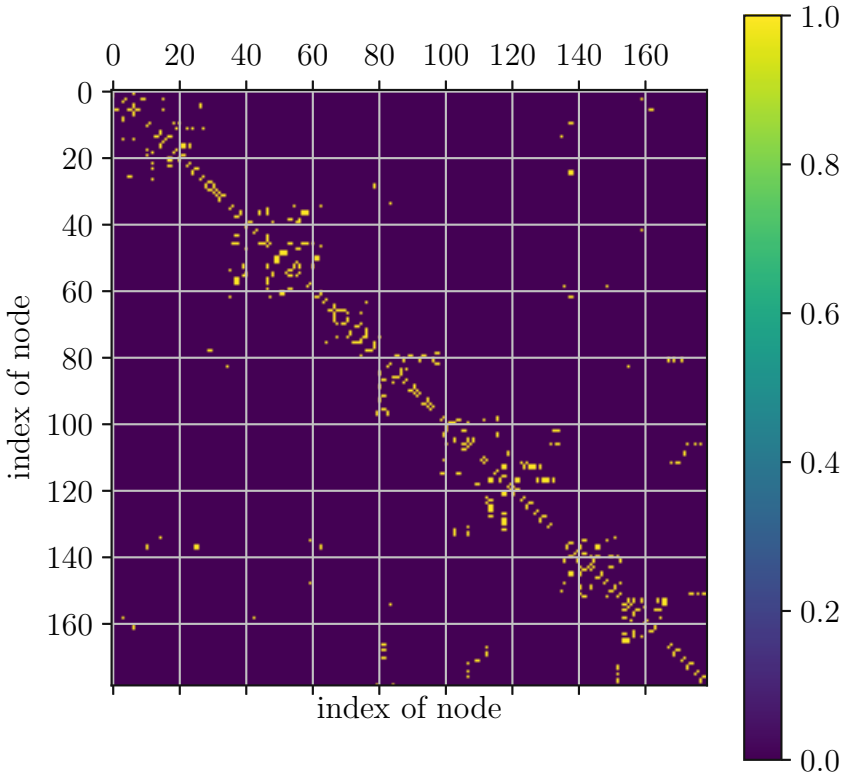

Figure 5.3: Encoding of the WECC-179 grid (cf. Fig. 5.2) as graph adjacency matrix.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

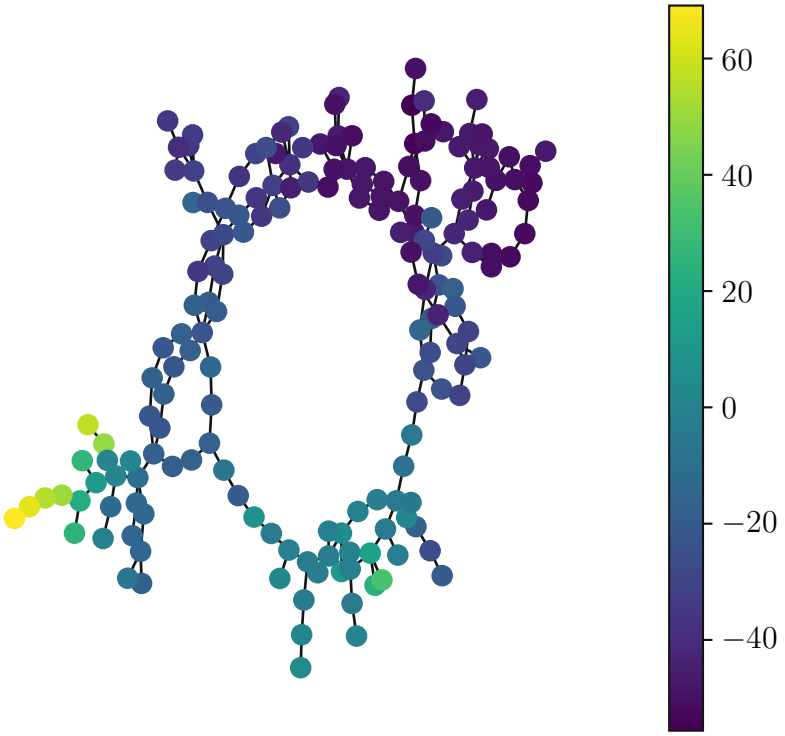

Figure 5.4: Voltage angle as graph signals for one fixed point in time. The angles are given in [°].

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

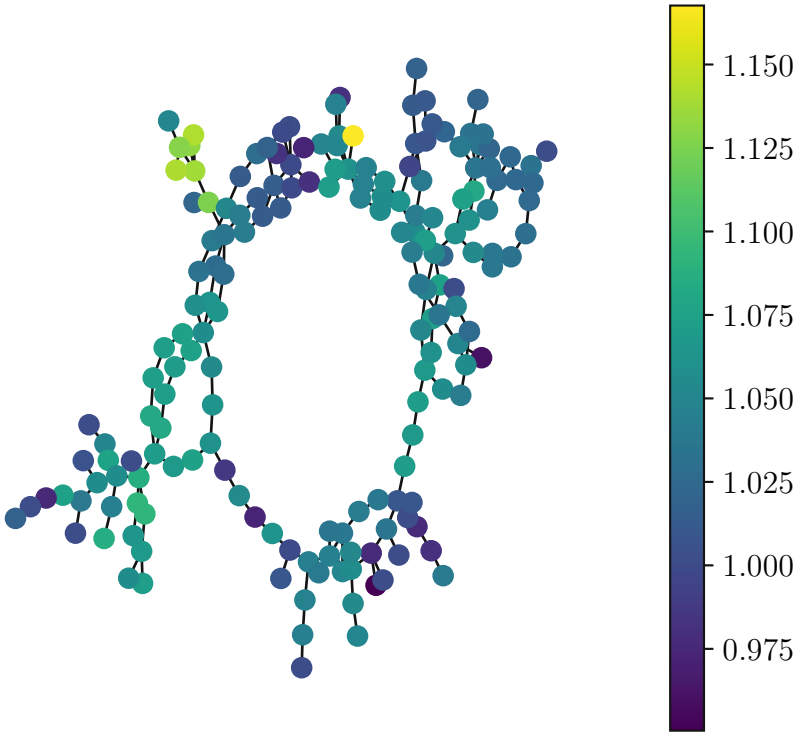

Figure 5.5: Voltage magnitude as graph signals for one fixed point in time. The magnitude is given in units of the nominal magnitude.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Chapter 6

Results

In order to compare the difference in performance for anomaly detection two models are created and trained. The first model uses the original GCN layer as given by Eq. (3.28) and Eq. (3.29). The second model uses the adapted GCN layer from Eq. (4.2) and Eq. (4.3). Fig. 6.1 shows the evolution of the loss function Eq. (5.4) during training. It shows, that the original GCN layer quickly reaches a limit, where no improvement of performance is possible whereas the adapted layer shows continuously improved performance over a longer training period.

In [11] it is argued, that in Eq. (3.28) $\tilde{\mathbf{D}}^{-1/2}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-1/2}$ with $\tilde{\mathbf{A}} = \mathbf{A}+\mathbf{I}_N$ should be used instead of $\mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}$ because otherwise training might become unstable due to exploding or vanishing gradients. In the adapted GCN Eq. (4.2) it is necessary to use $\mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}$. During training, no instabilities are observed. Fig. 6.5 shows the evolution of weights and biases for both GCN layers during training. It can be seen, that during the first epochs of training the weights and biases are change by rather large updates. During the course of the training, the speed of change in weights and biases decrease. This is a good visual hint, that the deep neural network converges towards a stable local optimum during training.

The model performance always deviates a little based on the test data. To gain some insights into this deviation, the test data of 1200 samples is separated into 37 batches of each 32 samples. The performance is calculated for each of the 37 batches and box plots are for both models. Fig. 6.2 shows the significantly better performance of the adapted model.

Fig. 6.3 and Fig. 6.4 show confusion matrices for the two models. The performance of the model using the original layer is hardly above randomly predicting anomalies. The model using the adapted GCN layer has sound performance in detecting false data anomalies.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

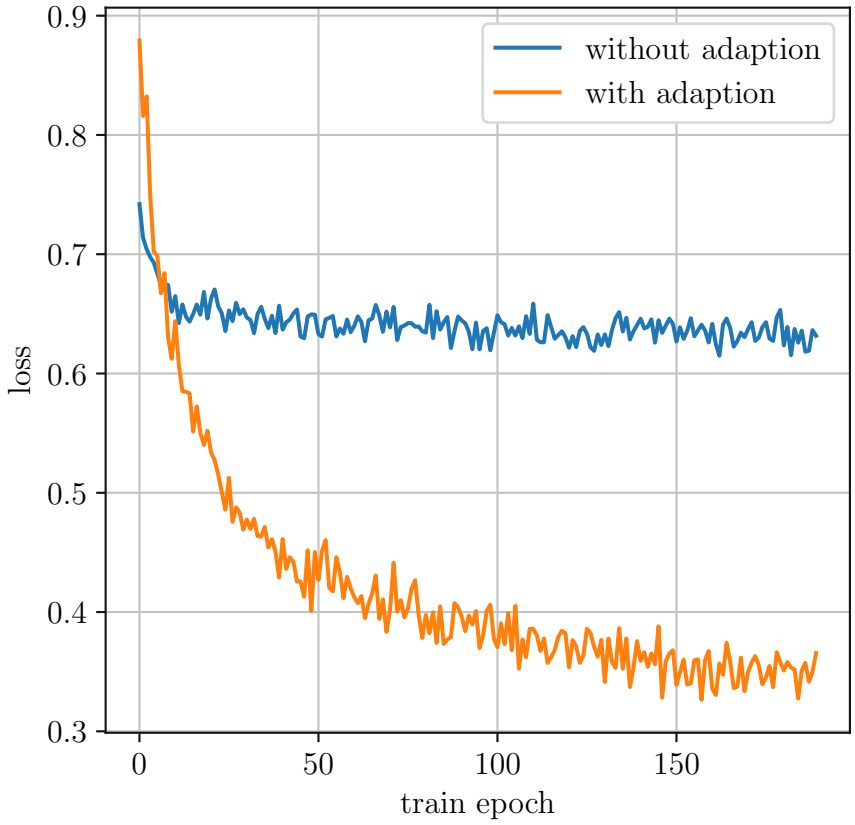

Figure 6.1: Comparison of loss function Eq. (5.4) during training for the original GCN layer Eq. (3.28) (in blue) and the adapted GCN layer Eq. (4.2) in orange. The model using the original layer soon reaches a level, where no improvement is possible because it can not adapt to the data. Adapted GCN shows continuos improvement of performance.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

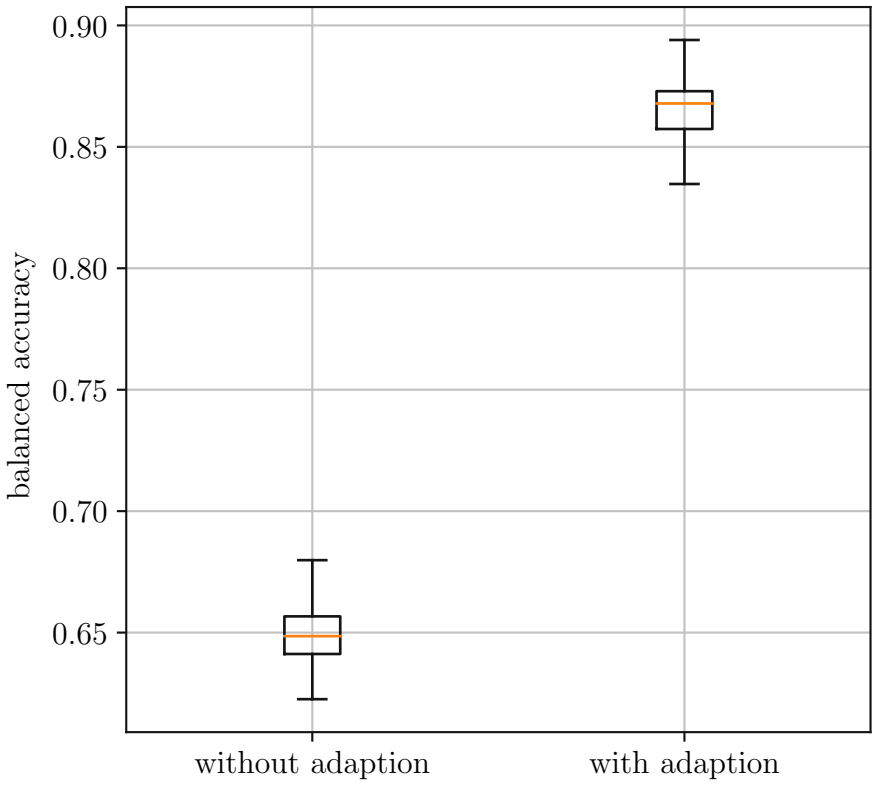

Figure 6.2: Model performance for different batches of test data. The performance of the model using adapted GCN layer is significantly better than the model using the original layer.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

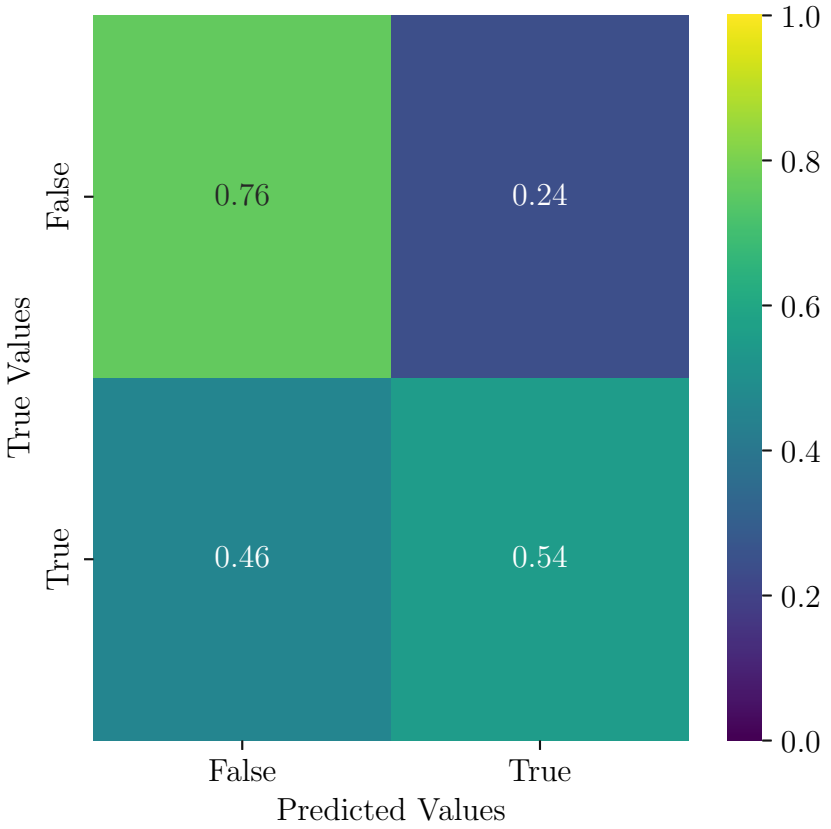

Figure 6.3: Confusion matrix for the model using the original GCN layer. *True* means anomaly, *False* means no anomaly.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

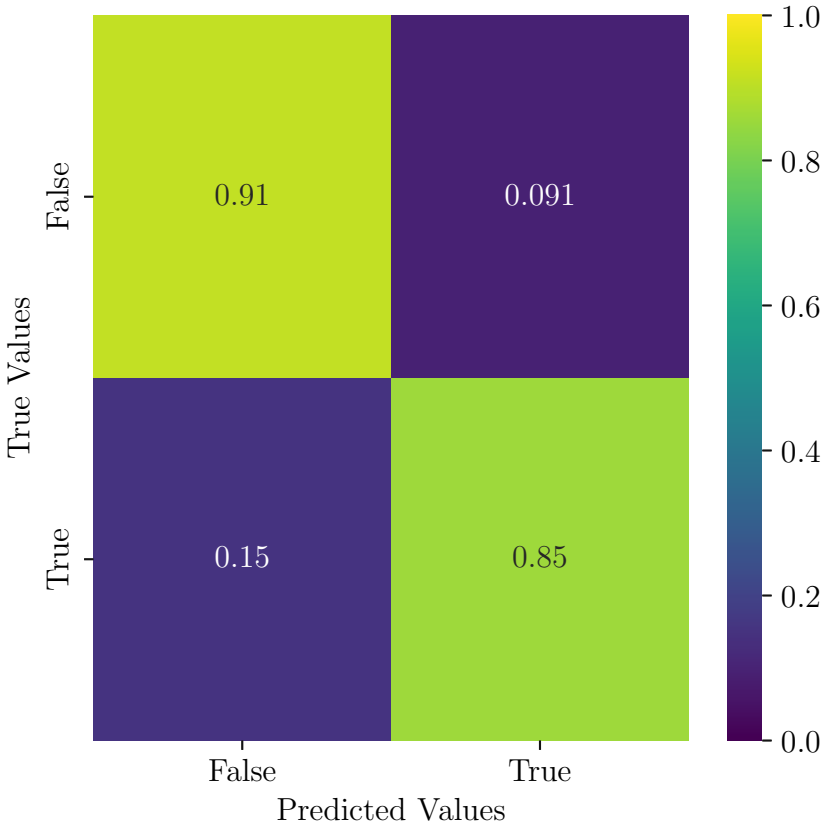

Figure 6.4: Confusion matrix for the model using the adapted GCN layer. *True* means anomaly, *False* means no anomaly.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

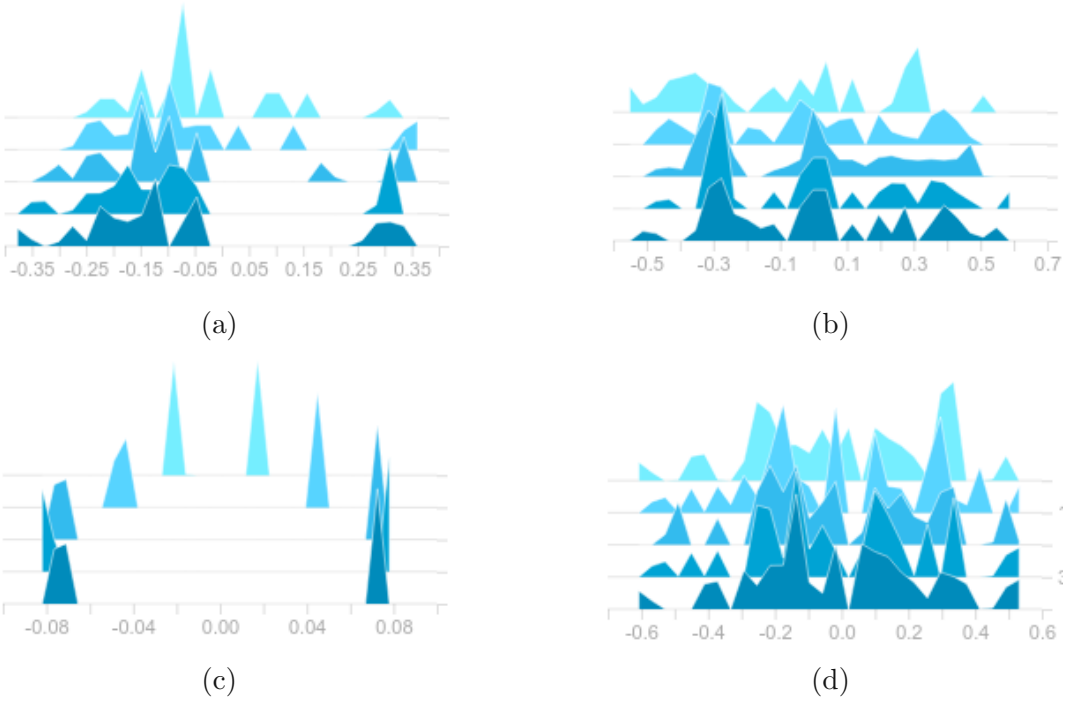

Figure 6.5: These images show the evolution of weights and biases during training, (a) for the weights of GCN layer 1, (b) for the biases of GCN layer 1, (c) for the weights of GCN layer 2 and (d) for the biases of GCN layer 2. Every 32 train epochs, a histogram is created, that shows the distribution of the according weights or biases. Each line in one the images represents one of these snapshots. The histograms more in the background (i.e. upper lines) represent the early training phase, lines in the foreground (i.e. lower lines) represent the later training phase. These figures were created by TensorBoard (see [58]).

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Chapter 7

Conclusion and outlook

7.1 Conclusion

The goal of this thesis is to detect false data injection anomalies in voltage data acquired by PMUs in a power grid. The literature review shows that, in order to achieve this, three major contributions need to be made. First, a graph neural network needs to process data from a power grid. Second, the anomaly detection needs to be formulated as a machine learning problem, that can be applied to a graph neural network. And third, required by the chosen resolution for the second problem, it is necessary to use a graph convolutional neural network, that can detect meaningful patterns in graph signals, that rapidly change within a node's neighborhood.

This thesis shows that graph convolutional neural networks can be used to analyze data from power grids. In particular, it shows that it is possible to detect a false data injection attack.

Moreover this thesis delivers a proof-of-concept implementation of an anomaly detection based on convolutional graph neural networks, a topic that received only little attention in literature so far.

If detecting these anomalies is implemented as a classification problem, the tendency of GCN to over-smooth is a major problem. This thesis proposes an adapted GCN layer to overcome the problem over-smoothing. Experiments show that this layer has superior performance to detected anomalies and is stable in training.

7.2 Outlook

Numerous topics are only treated in a marginal way in this thesis and should be examined in more detail.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

One such topic is that the performance of anomaly detection using convolutional graph neural networks should be systematically compared between a classification architecture and an autoencoder. This thesis only shows how an implementation using classification can be implemented.

In addition, should be experimented with different types of anomalies. This thesis uses only point anomalies, which represent only a relatively small part of scenarios of a false data injection attack. Many other scenarios are conceivable for how data in the power grid could be manipulated. Different types of anomalies can have strong implications for the applicability of machine learning models.

In this thesis graph signals are used which have two dimensions per node, namely magnitude and angle of voltage. In examples in the literature, much higher-dimensional graph signals are used, sometimes thousands of dimensions per node. In such a scenario, GCN can arguably play to its strengths even better. It should be investigated how it is possible to encode values captured in a power grid in such a way that high-dimensional graph signals are generated.

The experimental setup of this thesis uses a relatively small graph with a relatively small data set. This leads to the fact that a relatively short training is sufficient. This could potentially mask certain instabilities that would occur in a longer training. A comprehensive and systematic investigation of the stability of the layers used in the training should be performed.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Bibliography

[1] M. Fey and J. E. Lenssen, "Fast Graph Representation Learning with PyTorch Geometric," https://github.com/pyg-team/pytorch_geometric, 5 2019.

[2] ——, "Fast graph representation learning with pytorch geometric," *arXiv preprint arXiv:1903.02428*, 2019.

[3] D. Grattarola and C. Alippi, "Graph neural networks in tensorflow and keras with spektral," *arXiv preprint arXiv:2006.12138*, 2020.

[4] O. Ferludin, A. Eigenwillig, M. Blais, D. Zelle, J. Pfeifer, A. Sanchez-Gonzalez, S. Li, S. Abu-El-Haija, P. Battaglia, N. Bulut *et al.*, "Tf-gnn: graph neural networks in tensorflow," *arXiv preprint arXiv:2207.03522*, 2022.

[5] R. Ramakrishna and A. Scaglione, "Grid-graph signal processing (grid-gsp): A graph signal processing framework for the power grid," *IEEE Transactions on Signal Processing*, vol. 69, pp. 2725–2739, 2021.

[6] A. Phadke and J. Thorp, "History and applications of phasor measurements," in *2006 IEEE PES Power Systems Conference and Exposition*. IEEE, 2006, pp. 331–335.

[7] Q. F. Zhang, X. Luo, E. Litvinov, N. Dahal, M. Parashar, K. Hay, and D. Wilson, "Advanced grid event analysis at iso new england using phasorpoint," in *2014 IEEE PES General Meeting— Conference & Exposition*. IEEE, 2014, pp. 1–5.

[8] S. Maslennikov, B. Wang, Q. Zhang, E. Litvinov *et al.*, "A test cases library for methods locating the sources of sustained oscillations," in *2016 IEEE Power and Energy Society General Meeting (PESGM)*. IEEE, 2016, pp. 1–5.

[9] Y. Liu, P. Ning, and M. K. Reiter, "False data injection attacks against state estimation in electric power grids," *ACM Transactions on Information and System Security (TISSEC)*, vol. 14, no. 1, pp. 1–33, 2011.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

[10] P. Sen, G. Namata, M. Bilgic, L. Getoor, B. Galligher, and T. Eliassi-Rad, "Collective classification in network data," *AI magazine*, vol. 29, no. 3, pp. 93–93, 2008.

[11] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *arXiv preprint arXiv:1609.02907*, 2016.

[12] M. Cheung, J. Shi, O. Wright, L. Y. Jiang, X. Liu, and J. M. Moura, "Graph signal processing and deep learning: Convolution, pooling, and topology," *IEEE Signal Processing Mag.*, vol. 37, no. 6, pp. 139–149, 2020.

[13] F. Gama, E. Isufi, G. Leus, and A. Ribeiro, "Graphs, convolutions, and neural networks: From graph filters to graph neural networks," *IEEE Signal Processing Mag.*, vol. 37, no. 6, pp. 128–138, 2020.

[14] M. M. Bronstein, J. Bruna, Y. LeCun, A. Szlam, and P. Vandergheynst, "Geometric deep learning: going beyond euclidean data," *IEEE Signal Processing Magazine*, vol. 34, no. 4, pp. 18–42, 2017.

[15] Z. Li, N. Kovachki, K. Azizzadenesheli, B. Liu, A. Stuart, K. Bhattacharya, and A. Anandkumar, "Multipole graph neural operator for parametric partial differential equations," *Advances in Neural Information Processing Systems*, vol. 33, pp. 6755–6766, 2020.

[16] J. Xiong, Z. Xiong, K. Chen, H. Jiang, and M. Zheng, "Graph neural networks for automated de novo drug design," *Drug Discovery Today*, vol. 26, no. 6, pp. 1382–1393, 2021.

[17] A. Mayr, S. Lehner, A. Mayrhofer, C. Kloss, S. Hochreiter, and J. Brandstetter, "Boundary graph neural networks for 3d simulations," *arXiv preprint arXiv:2106.11299*, 2021.

[18] A. Deng and B. Hooi, "Graph neural network-based anomaly detection in multivariate time series," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 5, 2021, pp. 4027–4035.

[19] D. Shuman, S. Narang, P. Frossard, A. Ortega, and P. Vandergheynst, "The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains," *IEEE Signal Processing Mag.*, vol. 30, no. 3, pp. 83–98, 2013.

[20] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How powerful are graph neural networks?" *arXiv preprint arXiv:1810.00826*, 2018.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

[21] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and S. Y. Philip, "A comprehensive survey on graph neural networks," *IEEE Trans. Neural Networks and Learning Systems*, vol. 32, no. 1, pp. 4–24, 2020.

[22] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, "The graph neural network model," *IEEE transactions on neural networks*, vol. 20, no. 1, pp. 61–80, 2008.

[23] C. Gallicchio and A. Micheli, "Graph echo state networks," in *The 2010 international joint conference on neural networks (IJCNN)*. IEEE, 2010, pp. 1–8.

[24] Y. Li, D. Tarlow, M. Brockschmidt, and R. Zemel, "Gated graph sequence neural networks," *arXiv preprint arXiv:1511.05493*, 2015.

[25] H. Dai, Z. Kozareva, B. Dai, A. Smola, and L. Song, "Learning steady-states of iterative algorithms over graphs," in *International conference on machine learning*. PMLR, 2018, pp. 1106–1114.

[26] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," *arXiv preprint arXiv:1606.09375*, 2016.

[27] J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun, "Spectral networks and locally connected networks on graphs," *arXiv preprint arXiv:1312.6203*, 2013.

[28] M. Henaff, J. Bruna, and Y. LeCun, "Deep convolutional networks on graph-structured data," *arXiv preprint arXiv:1506.05163*, 2015.

[29] A. Micheli, "Neural network for graphs: A contextual constructive approach," *IEEE Transactions on Neural Networks*, vol. 20, no. 3, pp. 498–511, 2009.

[30] J. Atwood and D. Towsley, "Diffusion-convolutional neural networks," *Advances in neural information processing systems*, vol. 29, 2016.

[31] M. Niepert, M. Ahmed, and K. Kutzkov, "Learning convolutional neural networks for graphs," in *International conference on machine learning*. PMLR, 2016, pp. 2014–2023.

[32] S. Cao, W. Lu, and Q. Xu, "Deep neural networks for learning graph representations," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 30, no. 1, 2016.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

[33] D. Wang, P. Cui, and W. Zhu, "Structural deep network embedding," in *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, 2016, pp. 1225–1234.

[34] T. N. Kipf and M. Welling, "Variational graph auto-encoders," *arXiv preprint arXiv:1611.07308*, 2016.

[35] S. Pan, R. Hu, G. Long, J. Jiang, L. Yao, and C. Zhang, "Adversarially regularized graph autoencoder for graph embedding," *arXiv preprint arXiv:1802.04407*, 2018.

[36] K. Tu, P. Cui, X. Wang, P. S. Yu, and W. Zhu, "Deep recursive network embedding with regular equivalence," in *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, 2018, pp. 2357–2366.

[37] W. Yu, C. Zheng, W. Cheng, C. C. Aggarwal, D. Song, B. Zong, H. Chen, and W. Wang, "Learning deep network representations with adversarially regularized autoencoders," in *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, 2018, pp. 2663–2671.

[38] Y. Li, O. Vinyals, C. Dyer, R. Pascanu, and P. Battaglia, "Learning deep generative models of graphs," *arXiv preprint arXiv:1803.03324*, 2018.

[39] M. Simonovsky and N. Komodakis, "Graphvae: Towards generation of small graphs using variational autoencoders," in *International conference on artificial neural networks*. Springer, 2018, pp. 412–422.

[40] T. Ma, J. Chen, and C. Xiao, "Constrained generation of semantically valid graphs via regularizing variational autoencoders," *Advances in Neural Information Processing Systems*, vol. 31, 2018.

[41] N. De Cao and T. Kipf, "Molgan: An implicit generative model for small molecular graphs," *arXiv preprint arXiv:1805.11973*, 2018.

[42] A. Bojchevski, O. Shchur, D. Zügner, and S. Günnemann, "Netgan: Generating graphs via random walks," in *International conference on machine learning*. PMLR, 2018, pp. 610–619.

[43] Y. Seo, M. Defferrard, P. Vandergheynst, and X. Bresson, "Structured sequence modeling with graph convolutional recurrent networks," in *International conference on neural information processing*. Springer, 2018, pp. 362–373.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

[44] Y. Li, R. Yu, C. Shahabi, and Y. Liu, "Diffusion convolutional recurrent neural network: Data-driven traffic forecasting," *arXiv preprint arXiv:1707.01926*, 2017.

[45] A. Jain, A. R. Zamir, S. Savarese, and A. Saxena, "Structural-rnn: Deep learning on spatio-temporal graphs," in *Proceedings of the ieee conference on computer vision and pattern recognition*, 2016, pp. 5308–5317.

[46] B. Yu, H. Yin, and Z. Zhu, "Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting," *arXiv preprint arXiv:1709.04875*, 2017.

[47] S. Yan, Y. Xiong, and D. Lin, "Spatial temporal graph convolutional networks for skeleton-based action recognition," in *Thirty-second AAAI conference on artificial intelligence*, 2018.

[48] Z. Wu, S. Pan, G. Long, J. Jiang, and C. Zhang, "Graph wavenet for deep spatial-temporal graph modeling," *arXiv preprint arXiv:1906.00121*, 2019.

[49] V. P. Dwivedi, C. K. Joshi, T. Laurent, Y. Bengio, and X. Bresson, "Benchmarking graph neural networks," *arXiv preprint arXiv:2003.00982*, 2020.

[50] M. Zhang, Z. Cui, M. Neumann, and Y. Chen, "An end-to-end deep learning architecture for graph classification," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 32, no. 1, 2018.

[51] Z. Ying, J. You, C. Morris, X. Ren, W. Hamilton, and J. Leskovec, "Hierarchical graph representation learning with differentiable pooling," *Advances in neural information processing systems*, vol. 31, 2018.

[52] H. Gao and S. Ji, "Graph u-nets," in *international conference on machine learning*. PMLR, 2019, pp. 2083–2092.

[53] J. Lee, I. Lee, and J. Kang, "Self-attention graph pooling," in *International conference on machine learning*. PMLR, 2019, pp. 3734–3743.

[54] D. K. Hammond, P. Vandergheynst, and R. Gribonval, "Wavelets on graphs via spectral graph theory," *Applied and Computational Harmonic Analysis*, vol. 30, no. 2, pp. 129–150, 2011.

[55] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," *arXiv preprint arXiv:1207.0580*, 2012.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

[56] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the thirteenth international conference on artificial intelligence and statistics*. JMLR Workshop and Conference Proceedings, 2010, pp. 249–256.

[57] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[58] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin *et al.*, "Tensorflow: Large-scale machine learning on heterogeneous distributed systems," *arXiv preprint arXiv:1603.04467*, 2016.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN