technische universität
dortmund

fakultät für informatik

Master-Thesis

Convolutional Neural Networks auf Graphrepräsentationen von Bildern

Matthias Fey

18. Februar 2017

Gutachter:
Prof. Dr. Heinrich Müller
M.Sc. Jan Eric Lenssen

Lehrstuhl Informatik VII
Graphische Systeme
TU Dortmund

fakultät für
informatik

Inhaltsverzeichnis

1. Gedachter Inhalt

Einleitung: Motivation Aufbau der Arbeit

Grundlagen: Graphen, insbesondere planare Graphen Neuronale Netze (Was ist ein CNN, wie ist der Convolution Operator definiert, nicht lineare Aktivierungsfunktion)

Graphrepräsentationen von Bildern Grid Superpixel Superpixelalgorithmen Merkmalextraktion (Momente) Merkmalselektion (Cov, PCA)

Lernen auf Graphen: Stand der Forschung: Spatial vs Spectral

Spatial: Patchy Zentralität Canonical Labeling Übertragung auf planare Graphen <- EIGENER ANTEIL (z.B. Grid Spiral) Komplexität Vorteile (einfache Architektur)/Nachteile (keine direkte Nachbarschaftsberücksichtigung möglich, keine Graph Coarsening möglich, Vorverarbeitung ist recht teuer und muss Preprocessed werden weil man das nicht über Matrixoperationen ausdrücken kann)

Spectral: Laplacian, Fourier Transformation GCN und kGCN (weisfeiler Lehman) Übertragung auf planare Graphen (Adjazenzpartitionierung) <- EIGENER ANTEIL Pooling/Coarsening Komplexität Vorteile (z.B. Nachbarschaftsberücksichtigung/keine Ordnung nötig)/Nachteile (rotationsinvariant)

Deep Learning auf variabler Input-Menge (SPP)

Augmentierung von Graphen (ist das überhaupt möglich)

Realisierung (Experimente) und Evaluation Adam-Optimizer Sparse Tensors Vorstellung Datensätze (MNIST, PascalVOC, CIFAR-10, ImageNet) Tensorflow Dropout L2-Regularisierung

Zusammenfassung und Ausblick

2. Einleitung

$\mathbb{R}$ und $\mathbb{N}$ sind mathematische Symbole [1].

3. Spektrale Graphentheorie

- *Spektrum* eines Graphen zur Untersuchung seiner Eigenschaften
- *algebraische* oder *spektrale Graphentheorie* genannt
- als Spektrum eines Graphen bezeichnet man die (nach Größe geordnete) Folge der Eigenwerte λ seiner Adjazenzmatrix, d.h. $A \cdot x = \lambda x$ (x Eigenvektoren)

Algebraische Methoden sind sehr effektiv bei Graphen, die regulär und symmetrisch sind. Als *Schleife* wird in der Graphentheorie eine Kante bezeichnet, die einen Knoten mit sich selbst verbindet. Ein Graph ohne Schleifen wird *schleifenloser* Graph genannt.

Sei d_v der Grad eines Knotens v eines Graphen G. Der *Laplacian* $\mathcal{L}$ eines Graphen ohne Schleifen und Mehrfachkanten ist definiert als

$$\mathcal{L}(u,v) = \begin{cases} d_v, & \text{wenn } u = v, \\ -1, & \text{wenn } u \textit{ undv} \text{ adjazent}, \\ 0, & \text{sonst.} \end{cases} \tag{3.1}$$

Der Graph Laplacian ist eine Generalisierung des Laplacian auf einem Gitter.

Damit ist $\mathcal{L} = D - A$. $\mathcal{L}$ kann normalisiert werden über $\mathcal{L}_{\text{norm}} = T^{-\frac{1}{2}} L T^{-\frac{1}{2}}$, wobei T die Diagonalmatrix beschreibt mit $T(v,v) = d_v$. Für einen *isolierten* Knoten v, d.h. $d_v = 0$, gilt die Konvention $T^{-1}(v,v) = 0$. Ebenso lässt sich $\mathcal{L}_{\text{norm}}$ definieren als

$$\mathcal{L}_{\text{norm}}(u,v) = \begin{cases} 1, & \text{wenn } u = v \text{und } d_v \neq 0, \\ -\frac{1}{\sqrt{d_u d_v}}, & \text{wenn } u \textit{ undv} \text{ adjazent}, \\ 0, & \text{sonst.} \end{cases} \tag{3.2}$$

Wenn G k-regulär ist, d.h. $T = \text{diag}(k)$, dann gilt $\mathcal{L}_{\text{norm}} = I - \frac{1}{k} A$.

Da $\mathcal{L}$ symmetrisch ist, sind seine Eigenwerte alle ≥ 0 (d.h. $\mathcal{L}$ ist positiv-semidefinit). Jede Reihen- und Spaltensumme von $\mathcal{L}$ ist 0.

Einem gewichtetem ungerichterem Graph G kann eine Gewichtsfunktion $w : V \times V \to \mathbb{R}$ zugeschrieben werden, sodass $w(u,v) = w(v,u)$ und $w(u,v) \geq 0$. Falls $\{u,v\} \notin \mathcal{E}$, dann $w(u,v) = 0$. Damit sind ungewichtete Graphen nur ein Spezialfall bei dem alle Gewichte 0 oder 1 sind. Der Grad d_v eines Knoten v ist dann definiert als

$$d_v = \sum_u w(u, v). \tag{3.3}$$

Dann gilt

$$\mathcal{L} = \begin{cases} 1 - \frac{w(v,v)}{d_v}, & \text{wenn } u = v \text{und } d_v \neq 0, \\ -\frac{w(u,v)}{\sqrt{d_u d_v}}, & \text{wenn } u \; und v \text{ adjazent}, \\ 0, & \text{sonst.} \end{cases} \tag{3.4}$$

Bemerke, dass hier Schleifen nicht explizit ausgeschlossen werden!

Eine Verschrumpfung eines Graphen G kann beschrieben werden über zwei verschiedene Knoten u und v zu einem neuen Knoten v^* mit

$$w(x, v^*) = w(x, u) + w(x, v) \tag{3.5}$$

und

$$w(v^*, v^*) = w(u, u) + w(v, v) + 2w(u, v) \tag{3.6}$$

Mit $\lambda_G := \lambda_1$ für einen Graphen G, gilt für einen Graphen H der aus G verkleinert wurde

$$\lambda_G \leq \lambda_H \tag{3.7}$$

3.1. Spectral Graph Domain

- *Spectral Graph Domain*: Der Raum der Eigenfunktionen von $\mathcal{L}$
- Analogon (Nachbildung) einer *Fourier-Transformation* von Funktionen auf gewichteten Graphen

Eine beliebige Funktion $f : V \to \mathbb{R}$ kann als ein Vektor in $\mathbb{R}^n$ gesehen werden. Dies impliziert eine Ordnung auf den Knoten. Wir schreiben $f \in \mathbb{R}^n$ für Funktionen auf den Knoten eines Graphen und $f(m)$ für den Wert des $m$ten Knoten.

Dann gilt für eine beliebige Funktion $f \in \mathbb{R}^n$

$$\mathcal{L}f(x) = \sum_{x \;\; y} w(x, y) \cdot (f(x) - f(y)) \tag{3.8}$$

wobei die Summe über x y die Summierung über alle Knoten y beschreibt, die adjazent zu x sind.

Angenommen G ist als ein reguläres Gitter definiert der Breite und Höhe M Dann hat ein Knoten $v_{x,y}$ genau 4 Nachbarn mit Kantengewicht $\frac{1}{(\delta w)^2}$, bei dem δw die euklidsche Distanz zwischen zwei Gitterpunkten beschreibt.

Für eine Funktion $f : M \times M \to \mathbb{R}$ gilt dann:

$$\mathcal{L}f(x,y) = \frac{4f(x,y) - f(x+1,y) - f(x-1,y) - f(x,y+1) - f(x,y-1)}{(\delta w)^2} \tag{3.9}$$

Damit kann ein Signal f mit der Multiplikation mit $\mathcal{L}$ als eine Weiterpropagation von f unter der Berücksichtigung der lokalen Nachbarn verstanden werden (*5-point Stencil*, d.h. $\mathcal{L}f \approx -\nabla^2 f$).

3.2. Diskrete Fourier Transformation

$\mathcal{L}$ besitzt genau n orthogonal zueinander stehende Eigenvektoren $\{u_l\}_{l=1}^n \in \mathbb{R}^n$. Eigenvektoren u_i sind auf 1 normiert, d.h. $||u_i||_2 = 1$. Diese werden auch *Graph Fourier Modes* genannt. Diesen sind Eigenwete $\{\lambda_l\}_{l=1}^n \in \mathbb{R}$ zugeordnet, die die „Frequenzen“ bzw. das Spektrum des Graphen beschreiben oder visuell betrachtet die Ausdehnung des Raumes, den die Eigenvektoren aufspannen. Bemerke dass $\lambda_0 = 0$, da für den Eigenvektor $\vec{u_0} = (1, 1, \ldots, 1)^T$ gilt, dass $\mathcal{L}\vec{u_0} = 0$. $\mathcal{L}$ ist diagonalisierbar über $\mathcal{L} = U\Lambda U^T$, wobei $U = [u_1, \ldots, u_n] \in \mathbb{R}^{n \times n}$ die *Fourier Basis* und $\Lambda = \text{diag}([\lambda_0, \ldots, \lambda_n]) \in \mathbb{R}^{n \times n}$. Die *Fourier Transformation* eines Signals $x \in \mathbb{R}^n$ ist dann definiert als $\hat{x} = U^T x$ und die Inverse als $x = U\hat{x}$.

3.3. Convolution

Wir suchen einen Operator $x *_G y$, der eine Faltung zweier Eingangssignale x, y zu einem Ausgangssignal umleitet. x beschreibt dabei die Knotenattribute und y die Gewichte.

Wir definieren $*_G$ in der Fouier-Domäne als

$$x *_G g = U \cdot (U^T \cdot x \odot \hat{g}) \tag{3.10}$$

wobei $\odot(A, B) = (a_{ij} \cdot b_{ij})$ die elementweise Multiplikation bzw. das *Hadamard-Produkt*.

Das Hadamard-Produkt löst sich auf, wenn $\hat{g}$ als eine Diagonalmatrix repräsentiert wird. Dann gilt

$$x *_G g = U \begin{pmatrix} \hat{g}(\lambda_0) & \cdots & 0 \\ 0 & \cdots & \hat{g}(\lambda_n) \end{pmatrix} U^T x = U\hat{g}(\Lambda)U^T x \tag{3.11}$$

Dann beschreibt $\hat{g}(\Lambda) = \text{diag}(\theta)$ eine Gewichtsfunktion mit n Variablen, $\theta \in \mathbb{R}^n$. Damit ist die Faltung bzw. die Gewichtung abhängig von der Input-Größe n, was extrem schlecht ist.

3.3.1. Beispiel

Wir betrachten eine einfache 3×3 Adjazenzmatrix, d.h. $|\mathcal{V}| = n = 3$.

$$A = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix} \tag{3.12}$$

mit Diagonalmatrix $D = \text{diag}(1, 2, 1)$.
Der Laplacian $\mathcal{L} = D - A$ ist dann

$$\mathcal{L} = \begin{pmatrix} 1 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 1 \end{pmatrix} \tag{3.13}$$

Nun müssen die Eigenvektoren der Matrix und dessen Eigenwerte bestimmt werden, d.h. wir müssen das folgende Eigenwertproblem lösen

$$\mathcal{L} \cdot \vec{u} = \lambda \cdot \vec{u} \tag{3.14}$$

Wir erhalten 3 Eigenvektoren und Eigenwerte mit

$$\lambda_0 = 0, \vec{u}_0 = \frac{1}{\sqrt{3}} \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \approx \begin{pmatrix} 0.58 \\ 0.58 \\ 0.58 \end{pmatrix}, \lambda_1 = 1, \vec{u}_1 = \frac{1}{\sqrt{2}} \begin{pmatrix} -1 \\ 0 \\ 1 \end{pmatrix} \approx \begin{pmatrix} -0.71 \\ 0 \\ 0.71 \end{pmatrix}, \lambda_2 = 3, \vec{u}_2 = \frac{1}{\sqrt{6}} \begin{pmatrix} 1 \\ -2 \\ 1 \end{pmatrix} \approx \begin{pmatrix} 0.41 \\ -0.82 \\ 0.41 \end{pmatrix} \tag{3.15}$$

Dann sind U, Λ und U^T definiert als

$$U \approx \begin{pmatrix} 0.58 & -0.71 & 0.41 \\ 0.58 & 0 & -0.82 \\ 0.58 & 0.71 & 0.41 \end{pmatrix}, \Lambda = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 3 \end{pmatrix}, U^T \approx \begin{pmatrix} 0.58 & 0.58 & 0.58 \\ -0.71 & 0 & 0.71 \\ 0.41 & -0.82 & 0.41 \end{pmatrix} \tag{3.16}$$

Angenommen wir haben ein Signal $x = (100, 10, 1)^T$, dann ist der Wert dieses Signals transformiert in die Fourier Domäne definiert als $\hat{x} \approx (64.09, -70.00, 33.07)^T$. Führen wir $\hat{x}$ auf x mittels $U \cdot \hat{x}$ zurück, erhalten wir korrekterweise $x = (100, 10, 1)^T$.

3.4. Chebyshev Polynome

3.5. Probleme

Rotationsinvariant

4. Graph Convolutional Networks

$$H^{(l+1)} = f(H^{(l)}, A) \tag{4.1}$$

$$f(H^{(l)}, A) = \sigma(AH^{(l)}W^{(l)}) \tag{4.2}$$

$$D_{ii} = \sum_j A_{ij} \tag{4.3}$$

Für die Potenz $x \in \mathbb{R}$ einer Diagonalmatrix $D \in \mathbb{R}^{N \times N}$ gilt:

$$D^x = \begin{pmatrix} d_{11} & 0 & \cdots & 0 \\ 0 & d_{22} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & d_{nn} \end{pmatrix}^x = \begin{pmatrix} d_{11}^x & 0 & \cdots & 0 \\ 0 & d_{22}^x & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & d_{nn}^x \end{pmatrix} \tag{4.4}$$

4.1. Erweiterung für mehrere Kantenattribute

Graph Convolutional Networks berücksichtigen nur eine Adjazenzmatrix. Das bedeutet insbesondere, dass ein Graph nur über ein Kantenattribut verfügen kann. Das ist für ungewichtete Graphen die Markierung einer Kante ($a_{ij} \in \{0, 1\}$) oder für gewichte Graphen das Gewicht einer Kante ($a_{ij} \in \mathbb{R}^+$). Eine Menge von Kantenattributen kann über mehrere Adjazenzmatrizen definiert werden. Damit ist es ebenfalls möglich unterschiedliche Kanten für unterschiedliche Attribute zu definieren.

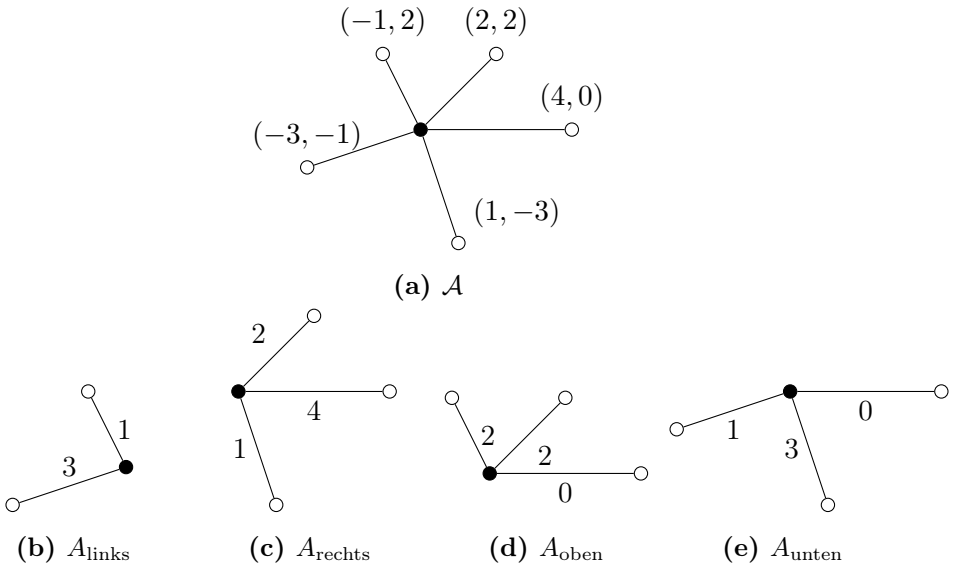
Eine Menge von Adjazenzmatrizen $\mathcal{A} = \{A_1, A_2, \ldots, A_m\}$ mit $A_i \in \mathbb{R}^{n \times n}$ beschreibt damit eine Menge von m Graphen über der gleichen Knotenmenge $\mathcal{V}$ mit Kardinalität n.

$\mathcal{A} \in \mathbb{R}^{m \times n \times n}$ kann zu einer zweidimensionalen Matrix $A \in \mathbb{R}^{m \cdot n \times n}$ geglättet werden.

Dann ist $A \cdot H^{(l)} \in \mathbb{R}^{m \cdot n \times d}$. Reshape zu $\mathbb{R}^{n \times m \cdot d}$ und Gewichtsmatrix $G \in \mathbb{R}^{m \cdot d \times x}$.

$$H^{(l+1)} = f(H^{(l)}, \tilde{\mathcal{A}}) = \sigma\left(\frac{1}{|\tilde{\mathcal{A}}|} \sum_{\tilde{A}_i \in \tilde{\mathcal{A}}} \tilde{D}_i^{-\frac{1}{2}} \tilde{A}_i \tilde{D}_i^{-\frac{1}{2}} H^{(l)} W_i^{(l)}\right) \tag{4.5}$$

$\sigma(\cdot)$ kennzeichnet eine Aktivierungsfunktion wie zum Beispiel $\mathrm{ReLU}(\cdot) = \max(0, \cdot)$.



(a) $\mathcal{A}$ **(b)** A_{links} **(c)** A_{rechts} **(d)** A_{oben} **(e)** A_{unten}

Abbildung 4.1.: Aufteilung einer Adjazenzmatrix in vier räumlich eingebettete Bereiche.

4.1.1. Übertragung auf räumlich eingebettete Graphen

Graphknoten haben im Allgemeinen keine Position oder Lage im Raum. Knoten, die Regionen in einer vorhandenen Segmentierung darstellen, haben jedoch offensichtlich eine gewisse Lage im Raum, die zum Beispiel über das Zentrum der Region definiert werden kann. Diese Information ist vorhanden und wichtig und sollte demnach auch nicht verloren gehen. Anstatt diese lokal im Knoten zu speichern, bietet es sich eher an diese Information in den Kanten zu speichern um eine bessere Faltung zu garantieren. Die euklische Distanz zwischen zwei benacharten Regionszentren wahrt zwar die Information der Distanz zweier Knoten zueinander, verliert aber die Information der Position zweier Knoten zueinander. Es bietet sich daher an, die horizontalen und vertikalen Abstände in einer Koordinate an den Kanten zu speichern. Es ist zu beachten, dass wir dadurch zu einem gerichteten Graphen übergehen, bei dem jede Kante von v nach w auch eine Kante von w nach v besitzt.

Wir haben damit zwei Adjazenzmatrizen. Da Graph Convolutional Networks nicht mit negativen Gewichten funktionieren, müssen wir negative Koordinaten in eine weitere Adjazenzmatrix schreiben. Wir gelangen damit zu vier Adjazenzmatrizen, die die Verbindungen von einem Knoten beschreibt, die links, rechts, oben oder unten zu ihm liegen. Wir definieren diese Adjazenzmatrizen respektive als $A_{\text{links}}, A_{\text{rechts}}, A_{\text{oben}}$ und A_{unten} (vgl. Abbildung 4.1). Falls eine Kante horizontal bzw. vertikal liegt, so definieren wir $a_{ij} = 1$ respektive für beide „gegenüberliegenden“ Adjazenzmatrizen.

Kantenattribute bzw. Positionen von Knoten sollten skalierungsinvariant gespeichert werden. Dafür werden die Abstände auf den Einheitskreis gemappt, wobei der Knoten mit

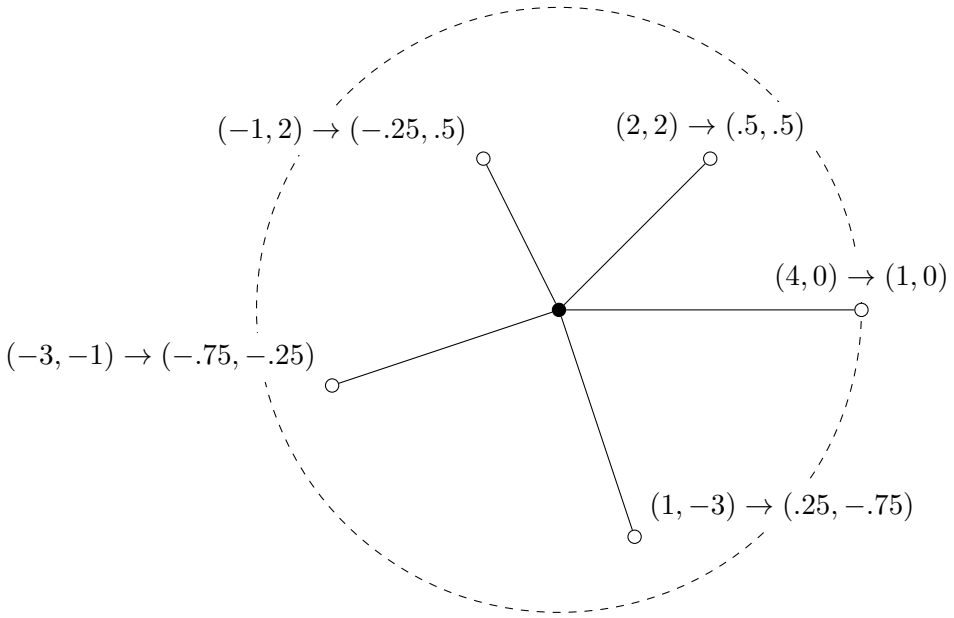


Abbildung 4.2.: Abbildung der lokalen Nachbarschaftsknoten auf den Einheitskreis.

der längsten Distanz zum Wurzelknoten genau auf dem Einheitskreis liegt (vgl. Abbildung 4.2).

Für die Anwendung auf das Graph Convolutional Network müssen die Gewichte aller Adjazenzmatrizen $a_{xij} \in [0, 1]$ invertiert werden, damit nähere Knoten einen größeren Einfluss haben. Ebenso müssen *Self Loops* für alle Knoten hinzugefügt werden. Wir definieren unsere Adjazenzmatrix $\tilde{A} \in \mathbb{R}^{N \times N}$ aus einer Adjazenzmatrix $A \in \mathbb{R}^{N \times N}$ dann über

$$\tilde{A}_{ij} = \begin{cases} 1, & \text{falls } i = j, \\ (a_{ij} + 1)^{-1}, & \text{falls } a_{ij} \neq 0, \\ 0, & \text{sonst.} \end{cases} \tag{4.6}$$

Dann ist $\tilde{a}_{ij} \in [1, 0.5]$

Diagonalmatrix ist schwierig. Man will ja die Normalisierung damit $H^{(l)}$ nicht überskaliert. Ich würde auch die gewichtete Matrix normalisieren. Denke das macht Sinn. Dann fallen die Werte ab, wenn viele Knoten weit entfernt sind.

A. Weitere Informationen

Symbolverzeichnis

$\mathbb{N}$ Menge der natürlichen Zahlen. 3

$\mathbb{R}^+$ Menge der positiven reellen Zahlen inklusive Null. 9

$\mathbb{R}$ Menge der reellen Zahlen. 3, 9

Abbildungsverzeichnis

Algorithmenverzeichnis

Literaturverzeichnis

[1] Nielsen, M. A.: *Neural Networks and Deep Learning.* Determination Press, 2015.

Eidesstattliche Versicherung

_____ _____

Name, Vorname Matr.-Nr.

Ich versichere hiermit an Eides statt, dass ich die vorliegende Bachelorarbeit/Masterarbeit* mit dem Titel

_____

_____

_____

selbstständig und ohne unzulässige fremde Hilfe erbracht habe. Ich habe keine anderen als die angegebenen Quellen und Hilfsmittel benutzt sowie wörtliche und sinngemäße Zitate kenntlich gemacht. Die Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

_____ _____

Ort, Datum Unterschrift

*Nichtzutreffendes bitte streichen

Belehrung:

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

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

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

Die oben stehende Belehrung habe ich zur Kenntnis genommen:

_____ _____

Ort, Datum Unterschrift