technische universität
dortmund

fakultät für informatik

Master-Thesis

Convolutional Neural Networks auf Graphrepräsentationen von Bildern

Matthias Fey

13. Juni 2017

Gutachter:

Prof. Dr. Heinrich Müller

M.Sc. Jan Eric Lenssen

Lehrstuhl Informatik VII
Graphische Systeme
TU Dortmund

Inhaltsverzeichnis

1 Einleitung **3**

1.1 Problemstellung 3

1.2 Aufbau der Arbeit 3

2 Grundlagen **5**

2.1 Mathematische Notationen 5

2.2 Graphentheorie 5

2.3 Convolutional Neural Networks 6

3 Graphrepräsentationen von Bildern **7**

3.1 Gitter 7

3.2 Superpixel 7

3.2.1 Verfahren 7

3.2.2 Adjazenzmatrixbestimmung 7

3.2.3 Merkmalsextraktion 7

4 Räumliches Lernen auf Graphen **9**

4.1 Räumliche Graphentheorie 9

4.2 Räumliche Faltung 9

4.3 Erweiterung auf Graphen im zweidimensionalen Raum 9

4.4 Netzarchitektur 9

5 Spektrales Lernen auf Graphen **11**

5.1 Spektrale Graphentheorie 11

5.1.1 Eigenwerte und Eigenvektoren reell symmetrischer Matrizen . 12

5.1.2 Laplace-Matrix 13

5.2 Spektraler Faltungsoperator 15

5.2.1 Graph-Fourier-Transformation 16

5.2.2 Spektrale Filterung 17

5.2.3 Polynomielle Approximation 18

5.3 Graph Convolutional Networks 20

5.4 Erweiterung auf Graphen im zweidimensionalen Raum 23
5.4.1 Partitionierung 25
5.4.2 Polynomielle Approximation über B-Spline-Kurven 27
5.5 Pooling auf Graphen 31
5.5.1 Graphvergröberung 31
5.5.2 Effizientes Pooling mittels binärer Bäumen 33
5.5.3 Erweiterung auf Graphen im zweidimensionalen Raum 35
5.6 Netzarchitektur 36

6 Evaluation **37**
6.1 Versuchsaufbau 37
6.1.1 Datensätze 37
6.1.2 Metriken 37
6.1.3 Parameterwahl 37
6.2 Merkmalsselektion 37
6.3 Ergebnisse 37
6.4 Laufzeitanalyse 38
6.5 Diskussion 38

7 Ausblick **39**

A Weitere Informationen **41**

Abbildungsverzeichnis **43**

Algorithmenverzeichnis **45**

Literaturverzeichnis **49**

Mathematische Notationen

$\|\cdot\|_2$ Skalarprodukt Menge? $(x_0, \ldots, x_n)$ geordnete Menge

1 Einleitung

Homepage[1]
Convolutional Neural Networks (CNNs) CNN $\mathbb{N}$
„wdawd“

1.1 Problemstellung

1.2 Aufbau der Arbeit

[1] https://github.com/rusty1s/embedded_gcnn

2 Grundlagen

2.1 Mathematische Notationen

Tensor, was bedeutet z.B. $\mathbf{W}_i$
Diagonalmatrix

2.2 Graphentheorie

Ein *Graph* $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ bezeichnet eine endliche Menge $\mathcal{V} = \{v_n\}_{n=1}^{N}$ von $N \in \mathbb{N}$ Knoten, $|\mathcal{V}| = N < \infty$, zusammen mit einer Menge geordneter Knotenpaare bzw. Kanten $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$. Seien $v_i, v_j \in \mathcal{V}$ im Folgenden zwei beliebige Knoten. Falls $(v_i, v_j) \in \mathcal{E}$, dann ist v_j *adjazent* zu v_i. Zu einem Graphen $\mathcal{G}$ definiert die *Nachbarschaftsfunktion* $\mathcal{N}(v_i) \subseteq \mathcal{V}$ über $\mathcal{N}(v_i) \coloneqq \{v_j \mid (v_i, v_j) \in \mathcal{E}\}$ die Nachbarschaftsmenge von v_i.
Ein *gewichteter Graph* $\mathcal{G} = (\mathcal{V}, \mathcal{E}, w)$ ist ein Graph, der zusätzlich eine *Gewichtsfunktion* $w\colon \mathcal{E} \to \mathbb{R}_+$ auf den Kanten des Graphen definiert, sodass $(v_i, v_j) \notin \mathcal{E}$ genau dann, wenn $w(v_i, v_j) = 0$. Im Falle eines ungewichteten Graphen ist die Gewichtsfunktion w implizit durch $\mathcal{E}$ über $w\colon \mathcal{E} \to \{0, 1\}$ gegeben.
Ein Graph heißt *ungerichtet*, falls $w(v_i, v_j) = w(v_j, v_i)$. Als *Schleife* wird eine Kante bezeichnet, die einen Knoten mit sich selbst verbindet, d.h. $w(v, v) > 0$ für einen Knoten $v \in \mathcal{V}$. Ein Graph ohne Schleifen wird *schleifenloser Graph* genannt. Für den weiteren Verlauf dieser Arbeit fordern wir, solange nicht explizit anders angegeben, gewichtete, ungerichtete sowie schleifenlose Graphen.
Ein Graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, w)$ kann weiterhin eindeutig über dessen (in der Regel dünnbesetzte) *Adjazenzmatrix* $\mathbf{A} \in \mathbb{R}_+^{N \times N}$ mit $\mathbf{A}_{ij} \coloneqq w(v_i, v_j)$ definiert werden. Als *Grad* eines Knotens $v \in \mathcal{V}$ wird die Anzahl der Knoten bezeichnet, die adjazent zu ihm sind, d.h. $\deg(v) \coloneqq |\mathcal{N}(v)|$. Im Falle von gewichteten Graphen wird der Grad eines Knotens von $v_i \in \mathcal{V}$ auch oft über $d(v_i) \coloneqq \sum_{j=1}^{N} \mathbf{A}_{ij}$ definiert. Die unterschiedliche Notation macht deutlich, wann wir welchen Grad eines Knotens meinen. Die *Gradmatrix* $\mathbf{D} \in \mathbb{R}_+^{N \times N}$ eines Graphen $\mathcal{G}$ ist dann definiert als Diagonalmatrix $\mathbf{D} \coloneqq \operatorname{diag}\left([d(v_1), \ldots, d(v_N)]^\top\right)$ bzw. $\mathbf{D}_{ii} = d(v_i)$. Ein Knoten $v \in \mathcal{V}$ heißt *isoliert*,

falls dieser keinen Nachbarsknoten besitzt, d.h. $\deg(v) = 0$.
Ein *Weg* der Länge K auf $\mathcal{G}$ ist eine Folge von Knoten $\left(v_{x(1)}, v_{x(2)}, \ldots, v_{x(K)}\right)$, sodass $\left(v_{x(k)}, v_{x(k+1)}\right) \in \mathcal{E}$ für alle $1 \leq k < K$, wobei $x\colon \{1, \ldots, K\} \to \{1, \ldots, N\}$ eine Abbildung auf die Indizes der Knoten $\{v_n\}_{n=1}^{N}$. Ein *Pfad* ist ein Weg mit der Bedingung, dass $v_{x(k)} \neq v_{x(k+1)}$. Im Kontext von schleifenlosen Graphen sind die Begriffe Weg und Pfad äquivalent. Wir schreiben $s(v_i, v_j)$ mit Hilfe der *Abstandsfunktion* $s\colon \mathcal{V} \times \mathcal{V} \to \mathbb{N} \cup \{\infty\}$ für die Länge des kürzesten Pfades von v_i nach v_j, d.h. die minimale Anzahl an Kanten, die zwischen v_i und v_j liegen. v_j ist von v_i aus *erreichbar*, wenn $s(v_i, v_j) \in \mathbb{N}$. Die Relation der Erreichbarkeit in der Knotenmenge $\mathcal{V}$ eines Graphen ist eine Äquivalenzrelation. Die Äquivalenzklassen der Erreichbarkeitsrelation heißen die *Zusammenhangskomponenten* des Graphen $\mathcal{G}$. Wir nennen $\mathcal{G}$ *zusammenhängend*, wenn er genau eine Zusammenhangskomponente besitzt, d.h. zu jedem Knoten v_i existiert mindestens ein Weg zu jedem anderen Knoten $v_j \in \mathcal{V}$. Es lässt sich weiter die Nachbarschaftsfunktion $\mathcal{N}$ zu einer *K-lokalisierten Nachbarschaftsfunktion* $\mathcal{N}_K \subseteq \mathcal{V}$ mit $\mathcal{N}_K(v_i) \coloneqq \{v_j | s(v_i, v_j) \leq K\}$ generalisieren.
Eine Funktion $f\colon \mathcal{V} \to \mathbb{R}$ auf den Knoten eines Graphen $\mathcal{G}$ heißt *Merkmalsfunktion*. Eine Merkmalsfunktion f kann ebenso als Vektor $\mathbf{f} \in \mathbb{R}^N$ mit $f(v_n) = \mathbf{f}_n$ geschrieben werden. Bildet f weiterhin auf eine Menge von Merkmalen $f\colon \mathcal{V} \to \mathbb{R}^M$, $M \in \mathbb{N}$, ab, so kann diese analog als eine *Merkmalsmatrix* $\mathbf{F} \in \mathbb{R}^{N \times M}$ mit $\mathbf{F}_{nm} = \left(f(v_n)\right)_m$ definiert werden.

2.3 Convolutional Neural Networks

conv2d CNN

3 Graphrepräsentationen von Bildern

$\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathrm{p})$
lokale oder globale Normierung
Ein *ebener Graph* ist eine konkrete Darstellung eines Graphen auf der zweidimensionalen Ebene $\mathbb{R}^2$. Jedem Knoten v ist eine Positionsfunktion $\mathrm{p}\colon \mathcal{V} \to \mathbb{R}^2$ zugeordnet, die die Position eines Knotens auf der Ebene eindeutig definiert.

3.1 Gitter

regulärer gittergraph (konnektivität)

3.2 Superpixel

$\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathrm{p}, \mathrm{m})$

3.2.1 Verfahren

SLIC. [2]
Simple Linear Iterative Clustering (SLIC)

Quickshift. [22]

Weitere Verfahren. [9]

3.2.2 Adjazenzmatrixbestimmung

3.2.3 Merkmalsextraktion

4 Räumliches Lernen auf Graphen

4.1 Räumliche Graphentheorie

Färbung von Knoten. awdawd

Isomorphie und kanonische Ordnung. awdawd

4.2 Räumliche Faltung

Knotenauswahl. awdawd

Nachbarschaftsgruppierung. awdawd

Normalisierung. awdawd

4.3 Erweiterung auf Graphen im zweidimensionalen Raum

4.4 Netzarchitektur

5 Spektrales Lernen auf Graphen

Das spektrale Lernen auf Graphen bzw. die Formulierung eines spektralen Faltungsoperator auf Graphen basiert auf der spektralen Graphentheorie, d.h. der Betrachtung des Spektrums eines Graphen definiert über dessen Eigenwerte. Merkmale auf den Knoten eines Graphen können über das Spektrum analog zur Fourier-Transformation in dessen Frequenzraum zerlegt und wieder retransformiert werden. Diese Transformation erlaubt damit die fundamentale Formulierung eines Faltungsoperators in der spektralen Domäne des Graphen. Da der so definierte spektrale Faltungsoperator insbesondere rotationsinvariant ist, wird dieser im Verlauf des Kapitels für den Kontext von Graphen im euklidschen Raum modifiziert.

Durch die spektrale Formulierung kann weiterhin ein effizientes Pooling auf Graphen formuliert werden, welches uns erlaubt, Netzarchitekturen auf Graphen völlig analog zu klassischen CNNs auf zweidimensionalen Bildern zu generieren.

5.1 Spektrale Graphentheorie

Die spektrale Graphentheorie beschäftigt sich mit der Konstruktion, Analyse und Manipulation von Graphen. Sie beweist sich dabei als besonders nützlich in Anwendungsgebieten wie der Charakterisierung von Expandergraphen, dem Graphenzeichnen oder dem spektralen Clustering (vgl. [20]). Weiterhin hat die spektrale Graphentheorie zum Beispiel auch Anwendungsgebiete in der Chemie, bei der die Eigenwerte des Spektrums des Graphen mit der Stabilität von Molekülen assoziert werden (vgl. [3]).

Es zeigt sich, dass die Eigenwerte des Spektrums eines Graphen eng mit den Eigenschaften eines Graphen verwandt sind. Als spektrale Graphentheorie versteht man damit insbesondere die Studie über die gemeinsamen Beziehungen dieser beiden Bereiche. Dieses Kapitel gibt eine Einführung in die wichtigsten Definitionen und Intuitionen der spektralen Graphentheorie, die es uns schlussendlich erlauben, die spektrale Faltung auf Graphen zu formulieren.

5.1.1 Eigenwerte und Eigenvektoren reell symmetrischer Matrizen

Das *Eigenwertproblem* einer Matrix $\mathbf{M} \in \mathbb{R}^{N \times N}$ ist definiert als $\mathbf{Mu} = \lambda \mathbf{u}$, wobei $\mathbf{u} \in \mathbb{R}^N$, $\mathbf{u} \neq \mathbf{0}$ *Eigenvektor* und $\lambda \in \mathbb{R}$ der entsprechende *Eigenwert* zu $\mathbf{u}$ genannt werden [12]. Ein Eigenvektor $\mathbf{u}$ beschreibt damit einen Vektor, dessen Richtung durch die Abbildung $\mathbf{Mu}$ nicht verändert, sondern lediglich um den Faktor λ skaliert wird. Zu einem Eigenwert λ gibt es unendlich viele (skalierte) Eigenvektoren $\mathbf{u}$. Wir definieren den Eigenvektor $\mathbf{u}$ eines Eigenwertes λ daher eindeutig über die Bedingung $\|\mathbf{u}\|_2 = 1$. Sei $\mathbf{M}$ weiterhin symmetrisch, d.h. $\mathbf{M} = \mathbf{M}^\top$ [12]. Dann gilt für zwei unterschiedliche Eigenvektoren $\mathbf{u}_1$ und $\mathbf{u}_2$, dass diese orthogonal zueinander stehen, d.h. $\mathbf{u}_1 \perp \mathbf{u}_2$, und $\mathbf{M}$ genau N reelle Eigenwerte mit $\{\lambda_n\}_{n=1}^N$ hat [12]. Wir definieren demnach zu $\mathbf{M}$ die orthogonale *Eigenvektormatrix* $\mathbf{U} \coloneqq [\mathbf{u}_1, \dots, \mathbf{u}_n] \in \mathbb{R}^{N \times N}$, d.h. $\mathbf{U}\mathbf{U}^\top = \mathbf{U}^\top\mathbf{U} = \mathbf{I}$, und dessen Eigenwertdiagonalmatrix $\mathbf{\Lambda} \coloneqq \operatorname{diag}\big([\lambda_1, \dots, \lambda_N]^\top\big)$, d.h. $\mathbf{\Lambda}_{ii} = \lambda_i$ [5]. Dann gilt $\mathbf{MU} = \mathbf{U\Lambda}$ und insbesondere ist $\mathbf{M}$ diagonalisierbar über [12]

$$\mathbf{M} = (\mathbf{MU})\mathbf{U}^\top = (\mathbf{U\Lambda})\mathbf{U}^\top.$$

Weiterhin gilt für die k-te Potenz von $\mathbf{M}$, $k \in \mathbb{N}$, [14]

$$\mathbf{M}^k = \left(\mathbf{U\Lambda U}^\top\right)^k = \mathbf{U\Lambda}^k\mathbf{U}^\top. \tag{5.1}$$

auf Grund der Induktion ($k - 1 \to k$)

$$\left(\mathbf{U\Lambda U}^\top\right)^k = \left(\mathbf{U\Lambda U}^\top\right)^{k-1}\mathbf{U\Lambda U}^\top = \mathbf{U\Lambda}^{k-1}\mathbf{U}^\top\mathbf{U\Lambda U} = \mathbf{U\Lambda}^k\mathbf{U}^\top.$$

Falls $\mathbf{M}$ weiterhin *schwach diagonaldominant* ist, d.h.

$$\sum_{\substack{j=1 \\ j \neq i}}^{N} |\mathbf{M}_{ij}| \leq |\mathbf{M}|_{ii}, \tag{5.2}$$

und weiterhin $\mathbf{M}_{ii} \geq 0$ für alle $i \in \{1, \dots, N\}$, dann ist $\mathbf{M}$ *positiv semidefinit*, d.h. $\mathbf{x}^\top\mathbf{Mx} \geq 0$ für alle $\mathbf{x} \in \mathbb{R}^N$ [12]. Eigenwerte symmetrischer positiv semidefiniter Matrizen $\lambda_i \in \mathbb{R}_+$ sind positiv reell und es lässt sich folglich auf diesen eine Ordnung definieren mit $0 \leq \lambda_1 \leq \dots \leq \lambda_N \coloneqq \lambda_{\max}$ [12].

5.1.2 Laplace-Matrix

Die Laplace-Matrix ist in der spektralen Graphentheorie eine Matrix, die die Beziehungen der Knoten und Kanten eines beliebigen Graphen $\mathcal{G}$ in einer generalisierten und normalisierten Form beschreibt. Viele der Eigenschaften von $\mathcal{G}$ können durch die Eigenwerte ihrer Laplace-Matrix beschrieben werden, wohingegen dies beispielsweise für die Eigenwerte der Adjazenzmatrix $\mathbf{A}$ von $\mathcal{G}$ nur bedingt zutrifft und insbesondere nicht verallgemeinbar für beliebge Graphen ist [3]. Dies ist vor allem dem Fakt geschuldet, dass die Eigenwerte der Laplace-Matrix konsistent sind mit den Eigenwerten des Laplace-Beltrami Operators ∇^2 in der spektralen Geometrie [3]. Die Laplace-Matrix ist damit ein geeignetes Mittel zur Betrachtung und Analyse eines Graphen.
Für einen schleifenlosen, ungerichteteten, gewichtet oder ungewichteten Graphen $\mathcal{G}$ und dessen Adjazenzmatrix $\mathbf{A}$ mit Gradmatrix $\mathbf{D}$ ist die *kombinatorische Laplace-Matrix* $\mathbf{L}$ definiert als $\mathbf{L} \coloneqq \mathbf{D} - \mathbf{A}$ [3]. Die *normalisierte Laplace-Matrix* $\tilde{\mathbf{L}}$ ist definiert als $\tilde{\mathbf{L}} \coloneqq \mathbf{D}^{-1/2}\mathbf{L}\mathbf{D}^{-1/2}$ mit der Konvention, dass $\mathbf{D}_{ii}^{-1/2} = 0$ für isolierte Knoten $v_i \in \mathcal{V}$ in $\mathcal{G}$, d.h. $\mathbf{D}_{ii} = 0$ [3]. Daraus ergibt sich die elementweise Definition

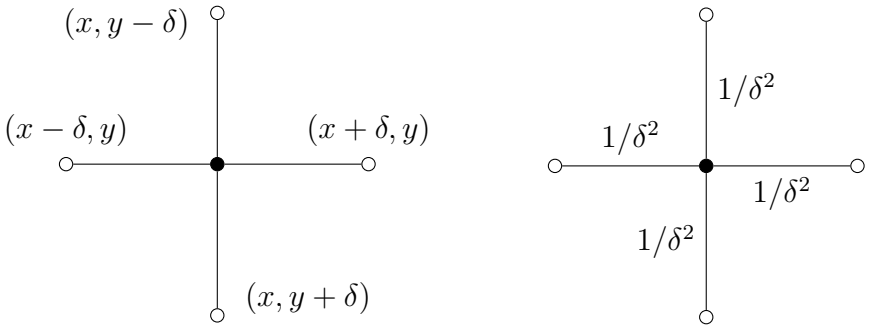
$$\tilde{\mathbf{L}}_{ij} \coloneqq \begin{cases} 1, & \text{wenn } i = j, \\ -\frac{w(v_i, v_j)}{\sqrt{d(v_i)d(v_j)}}, & \text{wenn } v_j \in \mathcal{N}(v_i), \\ 0, & \text{sonst.} \end{cases}$$

Für zusammenhängende Graphen kann $\tilde{\mathbf{L}}$ vereinfacht werden zu [3]

$$\tilde{\mathbf{L}} \coloneqq \mathbf{I} - \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}. \tag{5.3}$$

Jeder Eintrag auf der Diagonalen der normalisierten Laplace-Matrix ist folglich Eins. $\tilde{\mathbf{L}}$ ist damit normalisiert auf den (gewichteten) Grad zweier adjazenter Knoten v_i und v_j. Es ist anzumerken, dass $\mathbf{L}$ und insbesondere $\tilde{\mathbf{L}}$ symmetrisch sind, wohingegen eine Normalisierung der Form $\mathbf{D}^{-1}\mathbf{L}$ dies in der Regel nicht wäre [19]. $\mathbf{L}$ und $\tilde{\mathbf{L}}$ sind desweiteren keine ähnlichen Matrizen, insbesondere sind ihre Eigenvektoren verschieden. Die Nutzung von $\mathbf{L}$ oder $\tilde{\mathbf{L}}$ ist damit abhängig von dem Problem, welches man betrachtet [10]. Wir schreiben $\mathcal{L}$ wenn die Wahl der Laplace-Matrix, ob $\mathbf{L}$ oder $\tilde{\mathbf{L}}$, für die weitere Berechnung irrelevant ist.

Interpretation. Sei $f\colon \mathcal{V} \to \mathbb{R}$ bzw. $\mathbf{f} \in \mathbb{R}^N$ mit $f(v_i) = \mathbf{f}_i$ eine Funktion bzw. ein Signal auf den Knoten eines Graphen $\mathcal{G}$. Dann kann für die kombinatorische



(a) Gitterrepräsentation (b) Graphrepräsentation

Abbildung 5.1: Illustration des 5-Punkte-Sterns in zwei Dimensionen mit gleicher Approximation des ∇^2 Operators (bei umgekehrtem Vorzeichen), einmal mit der 5-Punkte-Stern Approximation auf regulären Gittern (a) und einmal mit der kombinatorischen Laplace-Matrix $\mathbf{L}$ auf Graphen (b).

Laplace-Matrix $\mathbf{L}$ verifiziert werden, dass sie die Gleichung

$$(\mathbf{Lf})_i = \sum_{v_j \in \mathcal{N}(v_i)} w(v_i, v_j)(\mathbf{f}_i - \mathbf{f}_j)$$

erfüllt [10]. Sei $\mathcal{G}$ nun ein Graph, der aus einem (unendlichen) zweidimensionalen regulärem Gitter entstanden ist, d.h. jeder Knoten v_i besitzt genau vier achsenparallele rechtwinklige Nachbarn mit gleichen Kantengewichten $1/\delta^2$, wobei $\delta \in \mathbb{R}$ den Abstand der Knoten zueinander beschreibt. Zur einfacheren Veranschaulichung benutzen wir dabei für die Signalstärke $\mathbf{f}_i$ eines Knoten v_i an Position (x, y) die Indexnotation $\mathbf{f}_{x,y}$. Dann beschreibt

$$(\mathbf{Lf})_{x,y} = \frac{4\mathbf{f}_{x,y} - \mathbf{f}_{x+1,y} - \mathbf{f}_{x-1,y} - \mathbf{f}_{x,y+1} - \mathbf{f}_{x,y-1}}{h^2}$$

die *5-Punkte-Stern* Approximation $\nabla^2 f$ (bei umgekehrtem Vorzeichen) definiert auf den Punkten $\{(x, y), (x+\delta, y), (x-\delta, y), (x, y+\delta), (x, y-\delta)\}$ [10] (vgl. Abbildung 5.1). Ähnlich zu einem regulären Gitter lässt sich ein Graph $\mathcal{G}$ auch über beliebig viele Abtastpunkte einer differenzierbaren Mannigfaltigkeit konstruieren. Es zeigt sich, dass mit steigender Abtastdichte und geeigneter Wahl der Kantengewichte die normalisierte Laplace-Matrix $\tilde{\mathbf{L}}$ zu dem kontinuierlichem Laplace-Beltrami Operator ∇^2 konvergiert [10]. Damit kann $\tilde{\mathbf{L}}$ als die diskrete Analogie des ∇^2 Operators auf Graphen verstanden werden. Der Laplace-Beltrami Operator $\nabla^2 f(p)$ misst dabei, in wie weit sich eine Funktion f an einem Punkt p von dem Durchschnitt aller Funktionspunkte um einen kleinen Bereich um p unterscheidet. Die Laplace-Matrix operiert dabei völlig analog, in dem sie misst, wie sehr sich eine (diskrete) Funktion

um einen Knoten im Vergleich zu seinen Nachbarknoten unterscheidet.
Die Eigenwerte und Eigenvektoren von $\mathcal{L}$ helfen uns beim Verständnis der linearen Transformation einer Funktion $\mathbf{f}$ (mehrfach) angewendet auf $\mathcal{L}$. Wir können dafür $\mathbf{f}$ als Linearkombination der Eigenbasis $\mathbf{f} = \sum_{n=1}^{N} a_n \mathbf{u}_n$, $a_n \in \mathbb{R}$ schreiben und erhalten

$$\mathcal{L}^k \mathbf{f} = \sum_{n=1}^{N} a_n \mathcal{L}^k \mathbf{u}_n = \sum_{n=1}^{N} a_n \lambda_n^k \mathbf{u}_n.$$

Somit können Eigenschaften von $\mathcal{L}$ und damit des Graphen selber durch dessen Eigenwerte und Eigenvektoren beschrieben werden.

Eigenschaften. $\mathcal{L} \in \mathbb{R}^{N \times N}$ ist eine reell symmetrische, positiv semidefinite Matrix [3]. Folglich besitzt $\mathcal{L}$ nach Kapitel 5.1.1 genau N positiv reelle Eigenwerte $\{\lambda_n\}_{n=1}^{N}$ mit Ordnung $0 \leq \lambda_1 \leq \cdots \leq \lambda_N$ und N korrespondierenden orthogonalen Eigenvektoren $\{\mathbf{u}_n\}_{n=1}^{N}$.
Die kombinatorische Laplace-Matrix $\mathbf{L}$ ist nach (5.2) weiterhin schwach diagonaldominant. Insbesondere summiert sich jede Reihen- und Spaltensumme von $\mathbf{L}$ zu Null auf, d.h. $\sum_{j=1}^{N} \mathbf{L}_{ij} = \sum_{j=1}^{N} \mathbf{L}_{ji} = 0$. Daraus folgt unmittelbar, dass $\lambda_1 = 0$, da $\mathbf{u}_1 = 1/\sqrt{N}[1, \ldots, 1]^\top \in \mathbb{R}^N$ Eigenvektor von $\mathbf{L}$ mit $\mathbf{L}\mathbf{u}_1 = \mathbf{0}$ [20]. $\tilde{\mathbf{L}}$ hingegen ist nicht zwingend schwach diagonaldominant. Es lässt sich jedoch zeigen, dass auch für $\tilde{\mathbf{L}}$ gilt, dass $\lambda_1 = 0$ [3].
Eine der interessantesten Eigenschaften eines Graphen ist dessen Konnektivität. Die Laplace-Matrix $\mathcal{L}$ bzw. deren Eigenwerte stellen ein geeignetes Mittel zur Untersuchung dieser Eigenschaft dar. So gilt beispielsweise für einen zusammenhängenden Graphen $\mathcal{G}$, dass $\lambda_2 > 0$. Falls $\lambda_i = 0$ und $\lambda_{i+1} \neq 0$, dann besitzt $\mathcal{G}$ genau i zusammenhängende Komponenten [3]. Damit ist die Anzahl der Null-Eigenwerte äquivalent zu der Anzahl an Komponenten, die ein Graph besitzt. Für $\tilde{\mathbf{L}}$ lässt sich weiterhin zeigen, dass $\lambda_{\max} \leq 2$ eine obere Schranke ihrer Eigenwerte ist [3].
Aus der Laplace-Matrix können ebenso Rückschlüsse über die kürzeste Pfaddistanz zweier Knoten gewonnen werden. So gilt für $\mathcal{L}^k$ mit $k \in \mathbb{N}$, dass $\mathcal{L}_{ij}^k = 0$ genau dann, wenn $s(v_i, v_j) > k$ [10]. Damit beschreibt $\mathcal{L}_i^k$ bildlich gesprochen die Menge an Knoten, die maximal k Kanten von v_i entfernt liegen.

5.2 Spektraler Faltungsoperator

Sei $\mathbf{f} \in \mathbb{R}^N$ ein Signal auf den Knoten eines Graphen $\mathcal{G}$, welches abhängig von der Struktur des Graphen weiter verarbeitet werden soll. Es ist jedoch nicht selbstver-

ständlich, wie recht einfache, dennoch fundamentale Signalverarbeitungsprozesse wie Translation oder Filterung und die daraus entstehende Faltung in der Domäne des Graphen definiert werden können [20]. So kann ein analoges Signal $f(t)$ beispielsweise mittels $f(t-3)$ um 3 nach rechts verschoben werden. Es ist hingegen völlig unklar, was es bedeutet, ein Graphsignal auf den Knoten um 3 nach rechts zu bewegen (vgl. [20]). Die spektrale Graphentheorie bietet uns dafür einen geeigneten Weg, indem Eingabesignale in das Spektrum des Graphen zerlegt bzw. abgebildet, modifiziert und wieder retransformiert werden können.

5.2.1 Graph-Fourier-Transformation

Das Spektrum eines Graphen $\mathcal{G}$ bilden die Eigenwerte $\{\lambda_n\}_{n=1}^N$ der Laplace-Matrix $\mathcal{L}$ von $\mathcal{G}$. Diese werden deshalb auch oft als die *Frequenzen* von $\mathcal{G}$ betitelt. In der spektralen Domäne können wir ein Eingabeignal $\mathbf{f}$ über $\mathcal{G}$ dann analog wie ein zeitdiskretes Abtastsignal in der Fourier-Domäne behandeln.

Klassische Fourier-Transformation. Die Fourier-Transformation $\hat{f}$ einer Funktion $f(t)$ ist definiert als [20]

$$\hat{f}(\omega) \coloneqq \left\langle f, e^{2\pi i\omega t} \right\rangle = \int_{\mathbb{R}} f(t) e^{-2\pi i\omega t} \, \mathrm{d}t.$$

Die komplexen Exponentiale $e^{2\pi i\omega t}$ beschreiben dabei die Eigenfunktionen des eindimensionalen Laplace-Beltrami Operators [20]:

$$-\nabla^2 e^{2\pi i\omega t} = -\frac{\partial^2}{\partial t^2} e^{2\pi i\omega t} = (2\pi\omega)^2 e^{2\pi i\omega t}. \tag{5.4}$$

$\hat{f}$ kann damit als die Ausdehnung von f in Bezug auf die Eigenfunktionen des Laplace-Beltrami Operators ∇^2 verstanden werden [10].

Analog lässt sich die *Graph-Fourier-Transformation* einer Funktion $f\colon \mathcal{V} \to \mathbb{R}$ bzw. $\mathbf{f} \in \mathbb{R}^N$ auf den Knoten eines Graphen $\mathcal{G}$ als Ausdehnung von f in Bezug auf die Eigenvektoren $\{\mathbf{u}_n\}_{n=1}^N$ der Laplace-Matrix $\mathcal{L}$ definieren [20]:

$$\hat{f}(\lambda_i) \coloneqq \langle \mathbf{f}, \mathbf{u}_i \rangle \qquad \text{bzw.} \qquad \hat{\mathbf{f}} \coloneqq \mathbf{U}^\top \mathbf{f}. \tag{5.5}$$

Die inverse Graph-Fourier-Transformation ergibt sich dann als [20]

$$f(v_i) = \sum_{n=1}^{N} \hat{f}(\lambda_n)(\mathbf{u}_n)_i \qquad \text{bzw.} \qquad \mathbf{f} = \mathbf{U}\hat{\mathbf{f}}. \tag{5.6}$$

In der klassischen Fourier-Analyse sind für die Eigenwerte $\left\{(2\pi\omega)^2\right\}_{\omega\in\mathbb{R}}$ in (5.4) nahe bei Null die korrespondieren Eigenfunktionen kleine, weich schwingende Funktionen, wohingegen für größere Eigenwerte bzw. Frequenzen die Eigenfunktionen sehr schnell und zügig anfangen zu oszillieren. Bei der Graph-Fourier-Transformation ist dies ähnlich. So ist für $\mathbf{L}$ der erste Eigenvektor $\mathbf{u}_1 = 1/\sqrt{N}[1,\ldots,1]^\top$ zum Eigenwert $\lambda_1 = 0$ konstant und an jedem Knoten gleich. Generell zeigt sich, dass die Eigenvektoren geringer Frequenzen nur geringfügig im Graph variieren, wohingegen Eigenvektoren größerer Eigenwerte immer unähnlicher werden (vgl. [20]).

Die Graph-Fourier-Transformation (5.5) und ihre Inverse (5.6) bieten uns eine Möglichkeit ein Signal in zwei unterschiedlichen Domänen zu repräsentieren, nämlich der Knotendomäne, d.h. das unveränderte Signal auf der Knotenmenge $f(v_i)$, und der spektralen Domäne, d.h. das transformierte Signal in das Spektrum des Graphen $\hat{f}(\lambda_i)$. Diese Transformation erlaubt uns die Formulierung fundamentaler Signalverarbeitungsoperationen.

5.2.2 Spektrale Filterung

In der Signalverarbeitung versteht man unter der Frequenzfilterung die Transformation eines Eingabesignals in die Fourier-Domäne und der verstärkenden oder dämpfenden Veränderung der Amplituden der Frequenzkomponenten. Formal betrachtet ergibt dies

$$\hat{f}_{\text{out}}(\omega) \coloneqq \hat{f}_{\text{in}}(\omega)\hat{g}(\omega) \tag{5.7}$$

mit dem Filter $\hat{g}\colon \mathbb{R} \to \mathbb{R}$. Shuman u. a. zeigen, dass die Filterung in der Fourier-Domäne äquivalent zu einer Faltung in der Zeitdomäne ist, d.h.

$$(f_{\text{in}} \star g)(t) \coloneqq \int_{\mathbb{R}} f_{\text{in}}(\tau)g(t-\tau)\,\mathrm{d}\tau = f_{\text{out}}(t). \tag{5.8}$$

Wir können die Filterung der Frequenzen in der Fourier-Domäne analog zu (5.7) für die spektrale Domäne auf Graphen über

$$\hat{f}_{\text{out}}(\lambda_i) \coloneqq \hat{f}_{\text{in}}(\lambda_i)\hat{g}(\lambda_i) \qquad \text{bzw.} \qquad \hat{\mathbf{f}}_{\text{out}} \coloneqq \hat{\mathbf{f}}_{\text{in}} \odot \hat{\mathbf{g}}$$

beschreiben, wobei $\odot$ das elementweise Hadamard-Produkt ist [20]. $\hat{\mathbf{g}} \in \mathbb{R}^N$ ist damit ein *nicht-parametrischer* Filter, d.h. ein Filter, dessen Werte für alle Frequenzen $\{\lambda_n\}_{n=1}^N$ frei wählbar sind [5]. Daraus ergibt sich analog zu (5.8) der *spektrale Faltungsoperator* auf Graphen in der Knotendomäne mit Hilfe der Graph-Fourier-Transformation (5.5) und ihrer Inversen (5.6) als [5, 20]

$$\mathbf{f}_{\text{in}} \star \hat{\mathbf{g}} \coloneqq \mathbf{U}\Big(\Big(\mathbf{U}^\top \mathbf{f}_{\text{in}}\Big) \odot \hat{\mathbf{g}}\Big) = \mathbf{f}_{\text{out}}. \tag{5.9}$$

5.2.3 Polynomielle Approximation

Es zeigt sich, dass die Benutzung des spektralen Faltungsoperators in (5.9) im Kontext eines CNNs auf Graphen mehrere Schwächen aufweist. So ist zum Beispiel die Auswertung von $\mathbf{f}_{\text{in}} \star \hat{\mathbf{g}}$ extrem berechnungsintensiv ist, denn die Multiplikation mit der dichtbesetzten Eigenvektormatrix $\mathbf{U}$ liegt in $\mathcal{O}(N^2)$ [5]. Zudem muss $\mathbf{U}$ zuerst bestimmt werden — ein kostspieliger Aufwand für Graphen mit möglicherweise weit mehr als hundert Knoten [14]. Desweiteren führt ein Filter $\hat{\mathbf{g}} \in \mathbb{R}^N$ der Größe N zu einem Lernaufwand in $\mathcal{O}(N)$, d.h. der Dimensionalität der Eingabedaten [5]. Ebenso kann $\hat{\mathbf{g}}$ so nicht für das Lernen auf Graphen mit variierendem N verwendet werden. Um die oben genannten Schwächen zu umgehen kann $\hat{g}(\lambda_i)$ über ein Polynom

$$\hat{g}(\lambda_i) \approx \sum_{k=0}^{K} c_k \lambda_i^k \tag{5.10}$$

vom Grad K mit Koeffizienten $[c_0, \ldots, c_K]^\top \in \mathbb{R}^{K+1}$ approximiert werden [5, 10]. Die Filtergröße sinkt somit auf einen konstanten Faktor K mit Lernaufwand $\mathcal{O}(K)$, dem gleichen Aufwand klassischer zweidimensionaler CNNs [5]. $\mathbf{f}_{\text{in}} \star \hat{\mathbf{g}}$ ergibt dann nach (5.1), (5.9) und (5.10) approximiert durch [5]

$$\mathbf{f}_{\text{in}} \star \hat{\mathbf{g}} \approx \sum_{k=0}^{K} c_k \mathbf{U}\mathbf{\Lambda}^k\mathbf{U}^\top \mathbf{f}_{\text{in}} = \sum_{k=0}^{K} c_k \mathcal{L}^k \mathbf{f}_{\text{in}}. \tag{5.11}$$

Insbesondere ist die spektrale Faltung damit nicht mehr abhängig von der Berechnung der Eigenwerte bzw. Eigenvektoren von $\mathcal{L}$. Mittels Kapitel 5.1.2 kann $\mathbf{f}_{\text{in}} \star \hat{\mathbf{g}}$ in der Knotendomäne nun als eine *lokaliserte lineare Transformation* interpretiert werden. So sammelt ein Summand $\mathcal{L}^k \mathbf{f}_{\text{in}}$ des spektralen Filters an einem Knoten v genau die Signale von Knoten auf, die maximal k Kanten von v entfernt liegen [10].

Eine Faltung über $f_{\text{in}}(v_i)$ wird damit als Linearkombination

$$f_{\text{out}}(v_i) \approx b_{ii} f_{\text{in}}(v_i) + \sum_{v_j \in \mathcal{N}_K(v_i)} b_{ij} f_{\text{in}}(v_j)$$

über dessen k-lokalisierte Nachbarschaft $\mathcal{N}_K(v_i)$ mit $b_{ij} \coloneqq \sum_{k=0}^{K} c_k \mathcal{L}_{ij}^k$ beschrieben [20].

Tschebyschow-Polynome. Obwohl der Aufwand zur Bestimmung von $\mathbf{f}_{\text{in}} \star \hat{\mathbf{g}}$ durch die polynomielle Approximation und insbesondere durch den Wegfall der Berechnung der Eigenvektormatrix $\mathbf{U}$ deutlich reduziert wurde, ist diese immer noch recht teuer aufgrund der Berechnung der K-ten Potenz von $\mathcal{L}$. So ist $\mathcal{L}$ zwar eine dünnbesetzte Matrix mit $|\mathcal{E}| + N \ll N^2$, $N \leq |\mathcal{E}|$, Einträgen, $\mathcal{L}^K$ ist dies jedoch zwangsläufig nicht. Eine Lösung zu diesem Problem ist die Benutzung eines Polynoms mit einer rekursiven Formulierung. Ein rekursives Polynom, dass dafür üblicherweise genutzt wird, ist das *Tschebyschow-Polynom*, da sich dieses zusätzlich durch einen sehr günstigen Fehlerverlauf auszeichnet (vgl. [10]). Tschebyschow-Polynome bezeichnen eine Menge von Polynomen $T_k(x) \colon \mathbb{R} \to \mathbb{R}$ mit dem rekursiven Zusammenhang

$$T_k(x) = 2x\, T_{k-1}(x) - T_{k-2}(x)$$

mit $T_0(x) = 1$ und $T_1(x) = x$ [10]. Ein Tschebyschow-Polynom T_k ist ein Polynom k-ten Grades und liegt im Intervall $[-1, 1]$ für $x \in [-1, 1]$ [10]. Wir können diese Tatsache nutzen und anstatt T_k auf $\mathbf{\Lambda} \in [0, \lambda_{\max}]^{N \times N}$ auf die skalierten und verschobenen Eigenwerte $\tilde{\mathbf{\Lambda}} \coloneqq 2\mathbf{\Lambda}/\lambda_{\max} - \mathbf{I} \in [-1, 1]^{N \times N}$ anwenden [5]. Die spektrale Faltung mittels Tschebyschow-Polynomen ergibt sich dann nach (5.11) als

$$\mathbf{f}_{\text{in}} \star \hat{\mathbf{g}} \approx \sum_{k=0}^{K} c_k \mathbf{U} T_k\Big(\tilde{\mathbf{\Lambda}}\Big) \mathbf{U}^\top \mathbf{f}_{\text{in}}, \tag{5.12}$$

wobei die Werte $[c_0, \ldots, c_K]^\top \in \mathbb{R}^{K+1}$ nun die Koeffizienten der Tschebyschow-Polynome $T_k\Big(\tilde{\mathbf{\Lambda}}\Big) \in \mathbb{R}^{N \times N}$ bilden [5]. $\mathbf{U} T_k\Big(\tilde{\mathbf{\Lambda}}\Big) \mathbf{U}^\top$ kann aufgrund der polynomiellen Form von T_k und (5.1) ebenso auf $T_k\Big(\tilde{\mathcal{L}}\Big)$ mit $\tilde{\mathcal{L}} \coloneqq \mathbf{U} \tilde{\mathbf{\Lambda}} \mathbf{U}^\top = \frac{2}{\lambda_{\max}} \mathcal{L} - \mathbf{I}$ angewendet werden [5]. Damit kann (5.12) weiter vereinfacht werden zu

$$\mathbf{f}_{\text{in}} \star \hat{\mathbf{g}} \approx \sum_{k=0}^{K} c_k T_k\Big(\tilde{\mathcal{L}}\Big) \mathbf{f}_{\text{in}}. \tag{5.13}$$

und ist nun ein rekursiv formulierter Faltungsoperator, der weiterhin ohne die explizite Berechnung von $\mathbf{U}$ auskommt [5]. Für $\mathcal{L}$ kann $\lambda_{\max} \leq 2$ auf dessen obere Schranke, d.h. $\lambda_{\max} \coloneqq 2$, gesetzt werden sodass die Berechnung von $\lambda_{\max}$ vermieden werden kann ohne die Schranken von $\tilde{\mathbf{\Lambda}} \in [-1, 1]^{N \times N}$ zu verletzen.
Der rekursive Zusammenhang von T_k hilft uns dabei, $\mathbf{f}_{\text{in}} \star \hat{\mathbf{g}}$ effizient zu bestimmen. Berechne dafür $\bar{\mathbf{f}}_k \coloneqq T_k\big(\tilde{\mathcal{L}}\big)\mathbf{f}_{\text{in}} \in \mathbb{R}^N$ für alle $k \in \{0, \ldots, K\}$ rekursiv mit $\bar{\mathbf{f}}_0 = \mathbf{f}_{\text{in}}$, $\bar{\mathbf{f}}_1 = \tilde{\mathcal{L}}\,\mathbf{f}_{\text{in}}$ und $\bar{\mathbf{f}}_k = 2\tilde{\mathcal{L}}\,\bar{\mathbf{f}}_{k-1} - \bar{\mathbf{f}}_{k-2}$. Dann ergibt sich $\mathbf{f}_{\text{out}} = \big[\bar{\mathbf{f}}_0, \bar{\mathbf{f}}_1, \ldots, \bar{\mathbf{f}}_K\big]\mathbf{c} \in \mathbb{R}^N$ mit $\mathbf{c} \coloneqq [c_0, c_1, \ldots, c_K]^\top \in \mathbb{R}^{K+1}$ (vgl. [10]). $\mathbf{f}_{\text{out}}$ lässt sich damit über $K + 1$ Multiplikationen einer dünnbesetzen Matrix mit einem Vektor und einer abschließenden Vektormultplikation beschreiben. Mit $N \leq |\mathcal{E}|$ ergibt dies eine finale Laufzeit der spektralen Faltung von $\mathcal{O}(K|\mathcal{E}|)$ [5].

Implementierung. Für gewöhnlich besteht eine CNN-Schicht nicht nur aus einem, sondern aus M_{in} vielen Signalen bzw. Merkmalen pro Knoten mit jeweils unterschiedlichen Filtern bzw. Gewichten pro Ein- und Ausgabekarte. In klassichen zweidimensionalen CNNs werden diese Merkmale auf M_{out} viele Merkmale abgebildet, in dem für jede Ausgabekarte über jede Eingabekarte gefaltet und dessen Ergebnisse sukzessive aufsummiert werden (vgl. Kapitel 2.3). Modellieren wir diesen Fall für den spektralen Faltungsoperator, dann erhalten wir rekursiv für eine Merkmalsmatrix $\mathbf{F}_{\text{in}} \in \mathbb{R}^{N \times M_{\text{in}}}$ gefaltet auf eine Merkmalsmatrix $\mathbf{F}_{\text{out}} \in \mathbb{R}^{N \times M_{\text{out}}}$ über den N Knoten eines Graphen $\mathcal{G}$ mittels des Filtertensors $\mathbf{W} \in \mathbb{R}^{(K+1) \times M_{\text{in}} \times M_{\text{out}}}$

$$\mathbf{F}_{\text{out}} \coloneqq \sum_{k=0}^{K} \bar{\mathbf{F}}_k,$$

wobei $\bar{\mathbf{F}}_k = \big(2\tilde{\mathcal{L}}\,\bar{\mathbf{F}}_{k-1} - \bar{\mathbf{F}}_{k-2}\big)\mathbf{W}_k \in \mathbb{R}^{N \times M_{\text{out}}}$ mit $\bar{\mathbf{F}}_0 = \mathbf{F}_{\text{in}}\mathbf{W}_0$ und $\bar{\mathbf{F}}_1 = \tilde{\mathcal{L}}\,\mathbf{F}_{\text{in}}\mathbf{W}_1$.

5.3 Graph Convolutional Networks

Kipf und Welling motivieren einen weiteren Ansatz zur Faltung auf Graphen, genannt *Graph Convolutional Network (GCN)*, der auf der Methodik des spekralen Faltungsoperators aus Kapitel 5.2 aufbaut und dabei wie eine „differenzierbare und parametrisierte Generalisierung des eindimensionalen Weisfeiler-Lehman Algorithmus auf Graphen“ fungiert [14].

Faltungsoperator. Sei $\mathbf{f}_{\text{in}} \star \hat{\mathbf{g}} \approx \sum_{k=0}^{K} c_k T_k\big(\tilde{\mathcal{L}}\big)\mathbf{f}_{\text{in}}$ der in (5.13) definierte spektrale Faltungsoperator mit $K = 1$. Dann ist $\mathbf{f}_{\text{in}} \star \hat{\mathbf{g}}$ eine lineare Funktion bezüglich $\mathcal{L}$

und damit eine lineare Funktion auf dem Spektrum des Graphen [14]. Mit $K = 1$ betrachtet der spektrale Faltungsoperator nur noch die lokale Nachbarschaft eines jeden Knotens (vgl. 5.2.3). Es ist anzumerken, dass dies in der Regel keinen Nachteil darstellt. So hat es sich bei gegenwärtigen „State-of-the-Art"-CNNs auf Bildern ebenfalls eingebürgert, nur noch über minimale 3×3 Filtergrößen zu falten und stattdessen Merkmale weit entfernterer Knoten über die mehrfache Aneinanderreihung der Faltungsschichten mittels tieferer Netze zu gewinnen (vgl. [11, 14, 21]). Unter dieser Restriktion vereinfacht sich $\mathbf{f}_{\text{in}} \star \hat{\mathbf{g}}$ zu

$$\mathbf{f}_{\text{in}} \star \hat{\mathbf{g}} \approx c_0\, \mathbf{f}_{\text{in}} + c_1 \Big(\frac{2}{\lambda_{\max}} \mathcal{L} - \mathbf{I} \Big) \mathbf{f}_{\text{in}} \tag{5.14}$$

mit zwei freien Parametern c_0 und c_1 [14]. Für $\tilde{\mathbf{L}}$ auf einem zusammenhängenden Graphen $\mathcal{G}$ gilt dann nach (5.3) und (5.14) weiter

$$\mathbf{f}_{\text{in}} \star \hat{\mathbf{g}} \approx c_0\, \mathbf{f}_{\text{in}} + c_1 \big(\tilde{\mathbf{L}} - \mathbf{I} \big) \mathbf{f}_{\text{in}} = c_0\, \mathbf{f}_{\text{in}} - c_1 \mathbf{D}^{-\frac{1}{2}} \mathbf{A} \mathbf{D}^{-\frac{1}{2}} \mathbf{f}_{\text{in}}, \tag{5.15}$$

wobei $\lambda_{\max} \coloneqq 2$ auf dessen obere Schranke gesetzt wird [14]. Um die Gefahr des Overfittings und die Anzahl an Berechnungen pro Schicht weiter zu beschränken, reduziert sich (5.15) mit einem einzigen Parameter $c \coloneqq c_0$ mit $c = -c_1$ zu [14]

$$\mathbf{f}_{\text{in}} \star \hat{\mathbf{g}} \approx c \big(\mathbf{I} + \mathbf{D}^{-\frac{1}{2}} \mathbf{A} \mathbf{D}^{-\frac{1}{2}} \big) \mathbf{f}_{\text{in}}.$$

Die skalierten Eigenwerte von $\tilde{\mathbf{\Lambda}}$ liegen auf Grund der Addition mit $\mathbf{I}$ nun im Intervall $[0, 2]$ (vgl. [14]). Demnach können wiederholte Anwendungen des Faltungsoperators zu „numerischen Instabilitäten und folglich zu explodierenden oder verschwindenen Gradienten" führen [14]. Kipf und Welling führen zur Behebung dieses Problems die folgende Renormalisierung durch: $\mathbf{I} + \mathbf{D}^{-1/2} \mathbf{A} \mathbf{D}^{-1/2} \to \tilde{\mathbf{D}}^{-1/2} \tilde{\mathbf{A}} \tilde{\mathbf{D}}^{-1/2}$ mit $\tilde{\mathbf{A}} \coloneqq \mathbf{A} + \mathbf{I}$ und $\tilde{\mathbf{D}}_{ii} \coloneqq \sum_{j=1}^{N} \tilde{\mathbf{A}}_{ij}$. Der entgültige Faltungsoperator des GCNs ergibt sich dann als

$$\mathbf{f}_{\text{in}} \star \hat{\mathbf{g}} \approx c\, \tilde{\mathbf{D}}^{-\frac{1}{2}} \tilde{\mathbf{A}} \tilde{\mathbf{D}}^{-\frac{1}{2}} \mathbf{f}_{\text{in}} \tag{5.16}$$

auf einem einzigen freien Parameter $c \in \mathbb{R}$.

Implementierung. Die Faltung des GCNs auf Merkmalsmatrizen lässt sich analog zur Tensorimplementierung des spektralen Faltungsoperators in Kapitel 5.2.3 beschreiben, mit dem Unterschied, dass wir aufgrund der Festlegung von $K = 1$ keinen Filtertensor, sondern lediglich eine Filtermatrix $\mathbf{W} \in \mathbb{R}^{M_{\text{in}} \times M_{\text{out}}}$ nutzen. Die Faltung einer Eingabemerkmalsmatrix $\mathbf{F}_{\text{in}} \in \mathbb{R}^{N \times M_{\text{in}}}$ auf eine Ausgabemerkmalsma-

Eingabe: Initiale Knotenfärbung $\mathbf{h}^{(0)} \in \mathbb{R}^N$
Ausgabe: Finale Knotenfärbung $\mathbf{h}^{(T)} \in \mathbb{R}^N$ nach T Durchläufen
$t \leftarrow 0$
repeat
  for $v_i \in \mathcal{V}$ **do**
    $\mathbf{h}_i^{(t+1)} \leftarrow \text{hash}\left(\sum_{v_j \in \mathcal{N}(v_i)} \mathbf{h}_j^{(t)}\right)$
  end for
  $t \leftarrow t + 1$
until Konvergenz

Algorithmus 5.1: Eindimensionaler Weisfeiler-Lehman Algorithmus auf einer initialen Knotenfärbung $\mathbf{h}^{(0)} \in \mathbb{R}^N$ eines Graphen $\mathcal{G}$ mit $v_i \in \mathcal{N}(v_i)$ [23]. Der Prozess der Verfärbung eines jeden Knotens v_i auf Basis der Farben seiner lokalen Nachbarsknoten wird solange wiederholt, bis diese konvergieren.

trix $\mathbf{F}_{\text{out}} \in \mathbb{R}^{N \times M_{\text{out}}}$ ergibt sich dann als

$$\mathbf{F}_{\text{out}} \coloneqq \tilde{\mathbf{D}}^{-\frac{1}{2}} \tilde{\mathbf{A}} \tilde{\mathbf{D}}^{-\frac{1}{2}} \mathbf{F}_{\text{in}} \mathbf{W} \tag{5.17}$$

mit Faltungsaufwand $\mathcal{O}(M_{\text{in}} M_{\text{out}} |\mathcal{E}|)$, weil $\tilde{\mathbf{A}} \mathbf{F}_{\text{in}}$ effizient mit der Multiplikation einer dünnbesetzten mit einer dichtbesetzten Matrix implementiert werden kann [14].

Beziehung zum Weisfeiler-Lehman Algorithmus. Der *eindimensionale Weisfeiler-Lehman Algorithmus* beschreibt eine weit-untersuchte Methode zur Knotenklassifizierung eines Graphen basierend auf einer initialen Färbung bzw. Merkmalsverteilung auf den Knoten eines Graphen $\mathcal{G}$, die unter anderem zur Bestimmung von Graphisomorphismen genutzt wird [7]. Basierend auf einer initialen Knotenfärbung $\mathbf{h}^{(0)} \in \mathbb{R}^N$ wird die Farbe eines jeden Knotens $v_i \in \mathcal{V}$ sukzessive mit Hilfe einer Hashfunktion $\text{hash}(\cdot)$ so angepasst, dass sie die vorangegangene Farbe des Knotens zusammen mit den Farben seiner lokalen Nachbarschaft repräsentiert. Dieser Prozess wiederholt sich solange, bis eine stabile Knotenfärbung gefunden wurde, d.h. die gefundene Färbung des Graphen konvergiert (vgl. Algorithmus 5.1).
Sei die Hashfunktion nun gegeben als eine differenzierbare, nicht-lineare Aktivierungsfunktion $\sigma \colon \mathbb{R} \to \mathbb{R}$, beispielsweise $\text{ReLU}(\cdot) \coloneqq \max(\cdot, 0)$, eines neuronalen Netzes. Dann ergibt sich eine Faltungsschicht des GCNs durch die Verkettung des Faltungsoperators mit anschließender Aktivierung als

$$\mathbf{h}_i^{(t+1)} = \sigma\left(\sum_{v_j \in \mathcal{N}(v_i)} \frac{1}{\sqrt{d_i d_j}} \mathbf{h}_j^{(t)} \mathbf{W}^{(}(t))\right),$$

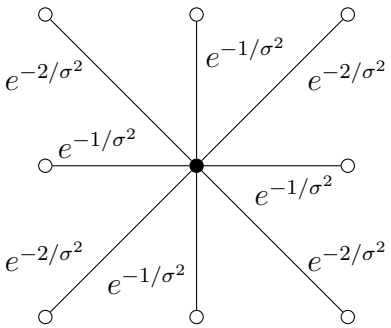
wobei $1/\sqrt{d_i d_j} \in \mathbb{R}$ die Normalisierungskonstante für die Kante $(v_i, v_j) \in \mathcal{E}$ entsprechend der Normalisierung $\tilde{\mathbf{D}}^{-1/2}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-1/2}$ aus (5.16) und $\mathbf{W}^{(t)} \in \mathbb{R}^{N \times N}$ die Filtermatrix der t-en Faltungsschicht beschreibt [14]. Folglich kann die Faltung des GCNs als „differenzierbare und parametrisierte Generalisierung des eindimensionalen Weisfeiler-Lehman Algorithmus auf Graphen“ verstanden werden [14].

5.4 Erweiterung auf Graphen im zweidimensionalen Raum

Die Ansätze von Defferrard u. a. aus Kapitel 5.2 und von Kipf und Welling aus Kapitel 5.3 zeigen konkurrenzfähige Resultate auf einer Reihe von Datensätzen auf Graphen (vgl. [5, 14]). So erreicht das GCN zum Beispiel in der teilweise-überwachten Knotenklassifizierung von *Referenzgraphen*, d.h. einer Menge von Knoten, die Dokumente über eine Reihe von Bag-of-Words-Merkmalen repräsentieren und (ungerichtet) über dessen Referenzierungen miteinander verbunden sind, beachtliche Ergebnisse und schneidet in diesen sogar knapp besser ab als über die Tschebyschow-Approximation mit $K = 2$ und $K = 3$ [14].
Die spektrale Faltung auf Graphen entspricht einer Generalisierung der Faltung klassischer CNNs auf zweidimensionalen Bildern [13]. Es ist jedoch anzumerken, dass die spektrale Faltung im Gegensatz zur klassischen Faltung auf einem regulären Gitter insbesondere rotationsinvariant ist. Das ist in der Regel für generelle Graphen keine Schwäche, schließlich kann den Knoten bzw. Kanten eines Graphen, kodiert als Adjazenzmatrix, keine Örtlichkeit bzw. Richtung (wie links, rechts, oben oder unten) zugeordnet werden. Die Rotationsinvarianz kann folglich sowohl als Einschränkung als auch als Vorteil interpretiert werden, abhängig von dem Problem, welches man betrachtet [5].
Im Kontext dieser Arbeit, dem Lernen auf Graphen im zweidimensionalen euklidschen Raum, bei denen Graphknoten eine eindeutige Position besitzen, ist die Rotationsinvarianz weitestgehend unerwünscht. Das kann leicht verifiziert werden, indem wir den Filter des GCNs auf einer Graphrepräsentation eines Gitters mit Abstand $\|1\|_2$ visualisieren (vgl. Abbildung 5.2). Diese Repräsentation entspricht damit genau dem Problem der zweidimensionalen Faltung auf Bildern mit einer Filtergröße von 3×3. Hier zeigt sich jedoch besonders deutlich die Limitierung des Netzes durch die Rotationsinvarianz. Wohingegen wir bei klassischen CNNs 3×3 unterschiedliche Parameter mit eindeutiger Örtlichkeit auf den benachbarten Bildpixeln trainieren, reduziert sich der Filter des GCNs (vereinfacht ohne Normalisierung mit $\tilde{\mathbf{D}}^{-1/2}$)

$(-1,-1)$	$(0,-1)$	$(1,-1)$
$(-1,0)$	$(0,0)$	$(1,0)$
$(-1,1)$	$(0,1)$	$(1,1)$

(a) Reguläres Gitter



(b) Graphrepräsentation

Abbildung 5.2: Illustration (a) eines 3×3 großen regulären Gitters zentriert um den Punkt $(0,0)$ und (b) dessen lokale Nachbarschaft der entsprechenden Graphrepräsentation mit einer Konnektivität von 8 bei horizontalen bzw. vertikalen Kantengewichten $\exp(-1/\sigma^2) \in \mathbb{R}$ bzw. $\exp(-2/\sigma^2) \in \mathbb{R}$ bei den Diagonalen.

effektiv zu einer Filtermatrix der Form

$$\begin{bmatrix} ce^{-2/\sigma^2} & ce^{-1/\sigma^2} & ce^{-2/\sigma^2} \\ ce^{-1/\sigma^2} & c & ce^{-1/\sigma^2} \\ ce^{-2/\sigma^2} & ce^{-1/\sigma^2} & ce^{-2/\sigma^2} \end{bmatrix} = c \begin{bmatrix} e^{-2/\sigma^2} & e^{-1/\sigma^2} & e^{-2/\sigma^2} \\ e^{-1/\sigma^2} & 1 & e^{-1/\sigma^2} \\ e^{-2/\sigma^2} & e^{-1/\sigma^2} & e^{-2/\sigma^2} \end{bmatrix}$$

mit einem einzigen trainierbaren Parameter $c \in \mathbb{R}$ bei horizontalen bzw. vertikalen Kantengewichten $\exp(-1/\sigma^2) \in \mathbb{R}$ bzw. $\exp(-2/\sigma^2) \in \mathbb{R}$ bei den Diagonalen. Damit reduziert sich das Training einer Faltungsschicht eines solchen GCNs letztendlich auf eine Skalarmultiplikation. Es scheint schwer vorstellbar mit diesem Ansatz komplexe Probleme wie zum Beispiel das Segmentieren eines Bildes zu lösen (vgl. [13]). Ein Vergleich zwischen der spektralen Faltung auf regulären Gittergraphen und der klassischen zweidimensionalen Faltung auf Bildern wird der spektralen Faltung aber nicht gerecht, schließlich wurden die klassischen CNNs speziell für die Anwendung auf Gittern entwickelt. So ist es zu erwarten, dass durch die Formulierung einer Faltung für generelle Graphen gewisse Einschränkungen in Kauf genommen werden müssen. Im Folgenden lässt sich der Faltungsoperator der GCNs aber insofern modifizieren, dass sich dieser für beliebige Graphen in einem zweidimensionalen euklischen Raum äquivalent zu der klassischen Formulierung auf regulären Gittern verhält.

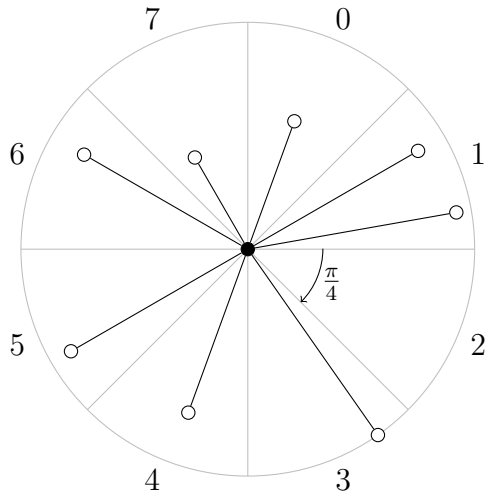


Abbildung 5.3: Partitionierung eines Graphknotens im Uhrzeigersinn in $P = 8$ Bereiche mit gleichmäßigen Innenwinkeln der Größe $\pi/4$.

5.4.1 Partitionierung

Sei $\mathcal{G}$ ein Graph im zweidimensionalen euklidschen Raum, eindeutig definiert über dessen Adjazenzmatrizen $\mathbf{A}_{\text{dist}} \in [0, 1)^{N \times N}$ und $\mathbf{A}_{\text{rad}} \in [0, 2\pi]^{N \times N}$ (vgl. Kapitel 3). Dann lässt sich $\mathcal{G}$ in $P \in \mathbb{N}$ Bereiche $\{\mathbf{A}_p\}_{p=0}^{P-1}$ *partitionieren*, sodass

$$(\mathbf{A}_p)_{ij} \coloneqq \begin{cases} (\mathbf{A}_{\text{dist}})_{ij}, & \text{wenn } (\mathbf{A}_{\text{rad}})_{ij} \in (2\pi p/P, 2\pi(p+1)/P] \\ 0, & \text{sonst.} \end{cases}$$

Damit beschreiben die Matrizen $\{\mathbf{A}_p\}_{p=0}^{P-1}$ disjunkte Partitionen der Kanten des Graphen $\mathcal{G}$ abhängig von ihren Ausrichtungen im Raum mit $\mathbf{A}_{\text{dist}} = \sum_{p=0}^{P-1} \mathbf{A}_p$. $\mathbf{A}_p$ ist insbesondere nicht symmetrisch, da $(\mathbf{A}_{\text{rad}})_{ij} \neq (\mathbf{A}_{\text{rad}})_{ji}$ für alle $v_i, v_j \in \mathcal{V}$ mit $v_j \in \mathcal{N}(v_i)$. Abbildung 5.3 veranschaulicht den Prozess der Partitionierung. Mit $P = 8$ erhalten die jeweiligen Bereiche zum Beispiel einen gleichmäßigen Innenwinkel der Größe $\pi/4$.

Faltungsoperator. Es lässt sich analog zu Kapitel 5.3 ein Faltungsoperator definieren, bei dem nun jeder Partition ein eigener frei trainierbarer Parameter zugeordnet wird. Der Parameter einer Partition hat damit folglich eine eindeutige Örtlichkeitszuweisung über das Intervall bzw. den Bereich der Richtungen seiner Kanten. Analog zu (5.16) muss dafür zuerst die (Re-)normalisierung der Form

$$\tilde{\mathbf{D}}_{\text{dist}}^{-\frac{1}{2}} \tilde{\mathbf{A}}_{\text{dist}} \tilde{\mathbf{D}}_{\text{dist}}^{-\frac{1}{2}} = \tilde{\mathbf{D}}_{\text{dist}}^{-1} + \sum_{p=0}^{P-1} \tilde{\mathbf{D}}_{\text{dist}}^{-\frac{1}{2}} \mathbf{A}_p \tilde{\mathbf{D}}_{\text{dist}}^{-\frac{1}{2}}$$

mit $\tilde{\mathbf{A}}_{\text{dist}} \coloneqq \mathbf{A}_{\text{dist}} + \mathbf{I}$ und $\left(\tilde{\mathbf{D}}_{\text{dist}}\right)_{ii} \coloneqq \sum_{j=1}^{N} \left(\tilde{\mathbf{A}}_{\text{dist}}\right)_{ij}$ durchgeführt werden. Dies lässt sich mit Hilfe von $\mathbf{A}_{\text{dist}} = \sum_{p=0}^{P-1} \mathbf{A}_p$ und $\tilde{\mathbf{D}}_{\text{dist}}^{-1/2}\tilde{\mathbf{D}}_{\text{dist}}^{-1/2} = \tilde{\mathbf{D}}_{\text{dist}}^{-1}$ verifizieren. Im Folgenden sei $\tilde{\mathbf{A}}_p \coloneqq \tilde{\mathbf{D}}_{\text{dist}}^{-1/2}\mathbf{A}_p\tilde{\mathbf{D}}_{\text{dist}}^{-1/2}$ für $p \in \{0, \dots, P-1\}$ und $\tilde{\mathbf{A}}_P \coloneqq \tilde{\mathbf{D}}_{\text{dist}}^{-1}$. Dann folgt für den Faltungsoperatur $\mathbf{f}_{\text{in}} \star \hat{\mathbf{g}}$ auf Graphen im zweidimensionalen euklidschen Raum, dass dieser über

$$\mathbf{f}_{\text{in}} \star \hat{\mathbf{g}} \approx \sum_{p=0}^{P} c_p \tilde{\mathbf{A}}_p \mathbf{f}_{\text{in}} \tag{5.18}$$

mit den freien Parametern $[c_0, \dots, c_P]^\top \in \mathbb{R}^{P+1}$ beschrieben werden kann.

Es stellt sich heraus, dass die Faltung in (5.18) auf den Partitionen eines regulären Gittergraphen mit $P = 8$ äquivalent zu der klassischen Faltung auf einem regulären Gitter mit Filtergröße 3×3 ist. Sei dafür $\mathcal{G}$ ein (unendlicher) regulärer Gittergraph bei einer Konnektivität von 8 und $\mathbf{f}_{\text{in}}$ Merkmalsvektor auf dem Graphen mit Koordinatenindexnotation $(\mathbf{f}_{\text{in}})_{x,y}$. Die klassische Faltung conv2d an einem Gitterpunkt (x, y) ist damit gegeben als

$$\text{conv2d}(\mathbf{f}_{\text{in}})_{x,y} = \sum_{i,j \in \{1,2,3\}} (\mathbf{f}_{\text{in}})_{x+i-2,y+j-2} \mathbf{W}_{i,j},$$

wobei $\mathbf{W} \in \mathbb{R}^{3\times 3}$ eine Filtermatrix der Größe 3×3 ist. Da $\tilde{\mathbf{A}}_{\text{dist}}$ ein reguläres Gitter beschreibt sind die Einträge ihrer Matrixreihen äquivalent unter unterschiedlicher Permutation und folglich sind die Einträge auf der Diagonalen von $\tilde{\mathbf{D}}_{\text{dist}}$ identisch (o.B.d.A. entfällt hier die Randknotenbetrachtung). Aufgrund der Partitionierung von $\mathbf{A}_{\text{dist}}$ in 8 disjunkte Bereiche $\left\{\tilde{\mathbf{A}}_p\right\}_{p=0}^{7}$ enthält $\tilde{\mathbf{A}}_p$ genau einen Eintrag pro Matrixreihe korrespondierend zu einer Kante des regulären Gitters. Für $p \in \{1, 3, 5, 7\}$ beschreibt $\tilde{\mathbf{A}}_p$ die horizontalen und vertikalen Kanten des Graphen mit jeweils gleichen Einträgen $\theta_0 \in \mathbb{R}$. Analog verweist $\tilde{\mathbf{A}}_p$ für $p \in \{0, 2, 4, 6\}$ auf die diagonalen Kanten des Graphen mit den festen Einträgen $\theta_1 \in \mathbb{R}$. Sei weiterhin o.B.d.A. $\theta_2 \coloneqq \left(\tilde{\mathbf{D}}_{\text{dist}}^{-1}\right)_{ii}$ für beliebiges $i \in \{0, \dots, N\}$. Mit der Zuordnung

$$\mathbf{W} = \begin{bmatrix} c_6\theta_1 & c_7\theta_0 & c_0\theta_1 \\ c_5\theta_0 & c_8\theta_2 & c_1\theta_0 \\ c_4\theta_1 & c_3\theta_0 & c_2\theta_1 \end{bmatrix}$$

für den Faltungsoperator $\mathbf{f}_{\text{in}} \star \hat{\mathbf{g}}$ aus (5.18) mit den Parametern $[c_0, \dots, c_8]^\top \in \mathbb{R}^9$ folgt damit sofort die Äquivalenz zu $\text{conv2d}(\mathbf{f}_{\text{in}})$ auf regulären Gittern.

Implementierung. Analog zu (5.17) lässt sich die Faltung für Merkmalsmatrizen $\mathbf{F}_{\text{in}} \in \mathbb{R}^{N \times M_{\text{in}}}$ und $\mathbf{F}_{\text{out}} \in \mathbb{R}^{N \times M_{\text{out}}}$ über einem Filtertensor $\mathbf{W} \in \mathbb{R}^{(P+1) \times M_{\text{in}} \times M_{\text{out}}}$ als

$$\mathbf{F}_{\text{out}} \coloneqq \sum_{p=0}^{P} \tilde{\mathbf{A}}_p \mathbf{F}_{\text{in}} \mathbf{W}_{p+1}$$

beschreiben. Es ist anzumerken, dass die Multiplikation mit den dünnbesetzten partitionierten Adjazenzmatrizen $\left\{\tilde{\mathbf{A}}_p\right\}_{p=0}^{P}$ extrem effizient ist, da $|\mathcal{E}_p| \ll |\mathcal{E}|$ und insbesondere $\sum_{p=0}^{P} |\mathcal{E}_p| = |\mathcal{E}| + N$ gilt, wobei $\mathcal{E}_p \in \mathcal{V} \times \mathcal{V}$ die Kantenmenge der Adjazenzmatrix $\tilde{\mathbf{A}}_p$ beschreibt. Die Laufzeit erhöht sich jedoch im Vergleich zu (5.17) durch die P-fache Multiplikation mit der Filtermatrix $\mathbf{W}_p$ zu $\mathcal{O}(PM_{\text{in}}M_{\text{out}}|\mathcal{E}|)$.

Obwohl die Faltung auf den Partitionen eines Graphen insbesondere durch die Äquivalenz zur klassischen Faltung auf regulären Gittern vielversprechend erscheint, gehen dabei dennoch Informationen über die Ausrichtung der Kanten im Raum verloren. Kanten mit verschiedenen Richtungen im Intervall $(2\pi p/P, 2\pi(p+1)/P]$ landen jeweils in der gleichen Partition p, auch wenn sich diese eventuell extrem unterscheiden. Eine Lösung zu diesem Problem ist sicherlich, P insoweit zu erhöhen, dass die Innenwinkel der einzelnen Partitionen entsprechend klein werden. Dies erscheint jedoch für beliebige, unbekannte Graphstrukturen nicht zwangsläufig sinnvoll. Neben dem erhöhten Aufwand der Faltung erhalten wir im schlimmsten Fall viele Partitionsmatrizen $\mathbf{A}_p$, die eine Nullmatrix $\mathbf{0}$ darstellen oder nur extrem wenige Kanten beinhalten. Kleinste Veränderungen in den Ausrichtungen der Kanten sorgen dann schließlich dafür, dass eine komplett andere Filtermatrix angesprochen wird. Ein dazu alternativ entwickelter Ansatz ist die Approximation des Filters über stückweise stetiger Polynome zwischen den Partitionsgrenzen des Graphen mit Hilfe von B-Spline-Kurven.

5.4.2 Polynomielle Approximation über B-Spline-Kurven

Eine B-Spline-Kurve beschreibt eine stückweise polynomielle Approximation einer Kurve vom Grad $K \in \mathbb{N}$ über $M+1$ Kontrollpunkten $\mathbf{d}_0, \ldots, \mathbf{d}_M \in \mathbb{R}^d$ [4]. Eine B-Spline Kurve kann dabei offen, d.h. mit beliebigen Endpunkten, sowie geschlossen, d.h. mit gleichem Anfangs- und Endpunkt, beschrieben werden [1]. Die *geschlossene B-Spline-Kurve* $b(t)$ ist auf dem Intervall $t \in (t_0, t_{M+1}]$ definiert als

$$b(t) \coloneqq \sum_{m=0}^{M} \mathbf{d}_m N_m^K(t), \tag{5.19}$$

mit den *B-Spline-Funktionen* $\left\{N_m^K\right\}_{m=0}^{M}$ zu einem *Knotenvektor* $\tau \in \mathbb{R}^{M+K+2}$ der Form $\tau \coloneqq [t_0, \ldots, t_{M+K+1}]^\top$ mit der Bedingung, dass $t_{M+k+2} \coloneqq t_{M+k+1} + (t_{k+1} - t_k)$ für $k \in \{0, \ldots, K-1\}$ [1]. Die B-Spline-Funktion $N_m^K \colon (t_0, t_{m+1}] \to [0,1]$ ist rekursiv über $k \in \{0, \ldots, K\}$ definiert mit der Initialisierung $(k = 0)$

$$N_m^0(t) \coloneqq \begin{cases} 1, & \text{falls } t \in (t_m, t_{m+1}], \\ 0, & \text{sonst} \end{cases}$$

und dem Rekursionsschritt $(k-1 \to k)$

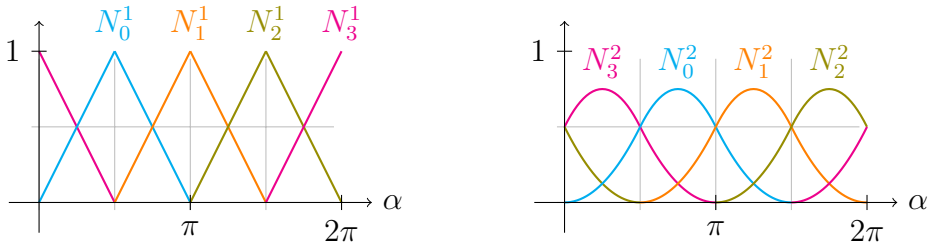
$$N_m^k(t) \coloneqq \frac{t - t_m}{t_{m+k} - t_m} N_m^{k-1}(t) + \frac{t_{m+k+1} - t}{t_{m+k+1} - t_{m+1}} N_{m+1}^{k-1}(t)$$

für $t \in (t_m, t_{M+1}]$ sowie $N_m^k(t) \coloneqq N_m^k(t - t_0 + t_{M+1})$ für $t \in (t_0, t_m]$ [1]. Ein Kontrollpunkt $\mathbf{d}_m$ beeinflusst die Kurve damit lediglich im Intervall $t_m < t \leq t_{m+K+1}$. Die Größe von K wird deshalb auch oft *lokale Kontrollierbarkeit* genannt. Weiterhin gilt für die Aufsummierung der B-Spline-Funktionen $\left\{N_m^K\right\}_{m=0}^{M}$, dass $\sum_{m=0}^{M} N_m^K(t) = 1$ für beliebige $K \in \mathbb{N}$ und $t \in (t_0, t_{M+1}]$ [4].

Wir können die Definition der B-Spline-Kurve $b(t)$ aus (5.19) nutzen, um sie aufbauend auf Kapitel 5.4.1 in das Anwendungsgebiet eines stückweisen polynomiellen Filters $b(\alpha)$ auf den Richtungen bzw. Winkeln eines Graphen $\mathcal{G}$, beschrieben durch $\mathbf{A}_{\text{dist}}$ und $\mathbf{A}_{\text{rad}}$, zu übertragen. Sei dafür $b \colon [0, 2\pi] \to \mathbb{R}$ im Folgenden eine B-Spline-Kurve auf den Winkeln der Graphkanten mit

$$b(\alpha) \coloneqq \sum_{p=0}^{P-1} c_p N_p^K(\alpha),$$

wobei die freien Parameter $[c_0, \ldots, c_{P-1}]^\top \in \mathbb{R}^P$ aus (5.18) nun die Koeffizienten der B-Spline-Funktionen $\left\{N_p^K\right\}_{p=0}^{P-1}$ bilden. Insbesondere definieren wir $b(0) \coloneqq 0$, da der Winkel 0 in der Adjazenzmatrix $\mathbf{A}_{\text{rad}}$ weiterhin angeben soll, dass $(v_i, v_j) \notin \mathcal{E}$. Wir können uns $b(\alpha)$ damit weiterhin als eine P-fache Partitionierung von $\mathcal{G}$ auf Basis seiner Kantenausrichtungen vorstellen, mit dem Unterschied, dass $b(\alpha)$ nun eine lokale Kontrollierbarkeit inne hält. Kanten, die vorher zwar in einer gleichen Partition lagen, sich jedoch eventuell stark in ihrer Ausrichtung unterschieden, sind nun „eindeutig“ differenzierbar über ihre abweichenden Auswertungen von $\left\{N_p^K\right\}_{p=0}^{P-1}$ bzw. ihrer Anteile von $[c_0, \ldots, c_{P-1}]^\top$ an $b(\alpha)$ entsprechend ihrer unterschiedlichen Winkel. Abbildung 5.4 soll dabei das Konzept geschlossener B-Spline-Funktionen auf Winkeln veranschaulichen. Mit der expliziten Forderung gleichmäßig verteiler Knoten im



(a) $K = 1$ (b) $K = 2$

Abbildung 5.4: Illustration geschlossener B-Spline-Funktionen $N_p^K \colon (0, 2\pi] \to [0, 1]$ für $P = 4$ gleichmäßig verteilter Kontrollpunkte, einmal mit lokaler Kontrollierbarkeit $K = 1$ (a) und einmal mit $K = 2$ (b).

Intervallbereich ergibt sich dann der Knotenvektor $\tau \coloneqq [\alpha_0, \ldots, \alpha_{P+K}]^\top \in \mathbb{R}_+^{P+K+1}$ mit $\alpha_p \coloneqq 2\pi p/P$ und erfüllt damit insbesondere die Bedingungen aus (5.19).

Faltungsoperator. Es kann folglich analog zu (5.18) der Faltungsoperator $\mathbf{f}_{\text{in}} \star \hat{\mathbf{g}}$ auf Basis eines stückweisen polynomiellen Filters beschrieben werden. Mit $\tilde{\mathbf{A}}_{\text{dist}} \in \mathbb{R}^{N \times N}$ und $\tilde{\mathbf{D}}_{\text{dist}} \in \mathbb{R}^{N \times N}$ ergibt sich $\mathbf{f}_{\text{in}} \star \hat{\mathbf{g}}$ dann als

$$\mathbf{f}_{\text{in}} \star \hat{\mathbf{g}} \approx c_P \tilde{\mathbf{D}}_{\text{dist}}^{-1} \mathbf{f}_{\text{in}} + \sum_{p=0}^{P-1} \left(\left(c_p N_p^K(\mathbf{A}_{\text{rad}}) \right) \odot \left(\tilde{\mathbf{D}}_{\text{dist}}^{-\frac{1}{2}} \tilde{\mathbf{A}}_{\text{dist}} \tilde{\mathbf{D}}_{\text{dist}}^{-\frac{1}{2}} \right) \right) \mathbf{f}_{\text{in}}$$

mit den freien Parametern $[c_0, \ldots, c_P]^\top \in \mathbb{R}^{P+1}$, wobei N_p^K elementweise auf die Matrix $\mathbf{A}_{\text{rad}}$ angewendet wird. $\mathbf{f}_{\text{in}} \star \hat{\mathbf{g}}$ sammelt dabei die aktuellen Merkmale an den einzelnen Knoten über $c_P \tilde{\mathbf{D}}_{\text{dist}}^{-1} \mathbf{f}_{\text{in}}$ und aggreriert diese mit den Merkmalen der lokalen Nachbarschaft über den polynomiellen Filter $\left(c_p N_p^K(\mathbf{A}_{\text{rad}}) \right) \odot \left(\tilde{\mathbf{D}}_{\text{dist}}^{-1/2} \tilde{\mathbf{A}}_{\text{dist}} \tilde{\mathbf{D}}_{\text{dist}}^{-1/2} \right)$, der so die Länge und Richtungen der Kanten berücksichtigt. Für $K = 0$ ist die Faltung über den B-Splines äquivalent zu der Faltung über den Partitionen des Graphen aus (5.18) aufgrund der Rechteckfunktionen $\left\{ N_p^0 \right\}_{p=0}^{P-1}$.

Implementierung. Für Merkmalsmatrizen $\mathbf{F}_{\text{in}} \in \mathbb{R}^{N \times M_{\text{in}}}$ und $\mathbf{F}_{\text{out}} \in \mathbb{R}^{N \times M_{\text{out}}}$ kann die Faltung über einem Filtertensor $\mathbf{W} \in \mathbb{R}^{(P+1) \times M_{\text{in}} \times M_{\text{out}}}$ damit als

$$\mathbf{F}_{\text{out}} \coloneqq \tilde{\mathbf{D}}_{\text{dist}}^{-1} \mathbf{F}_{\text{in}} \mathbf{W}_{P+1} + \sum_{p=0}^{P-1} \left(N_p^K(\mathbf{A}_{\text{rad}}) \odot \left(\tilde{\mathbf{D}}_{\text{dist}}^{-\frac{1}{2}} \tilde{\mathbf{A}}_{\text{dist}} \tilde{\mathbf{D}}_{\text{dist}}^{-\frac{1}{2}} \right) \right) \mathbf{F}_{\text{in}} \mathbf{W}_{p+1}$$

beschrieben werden. Im Vergleich zu Kapitel (5.4.1) erhöht sich der Aufwand der Faltung zu $\mathcal{O}(KPM_{\text{in}}M_{\text{out}})$, da die Partitionen der Adjazenzmatrizen nicht mehr disjunkt, sondern $(K + 1)$-mal betrachtet werden.

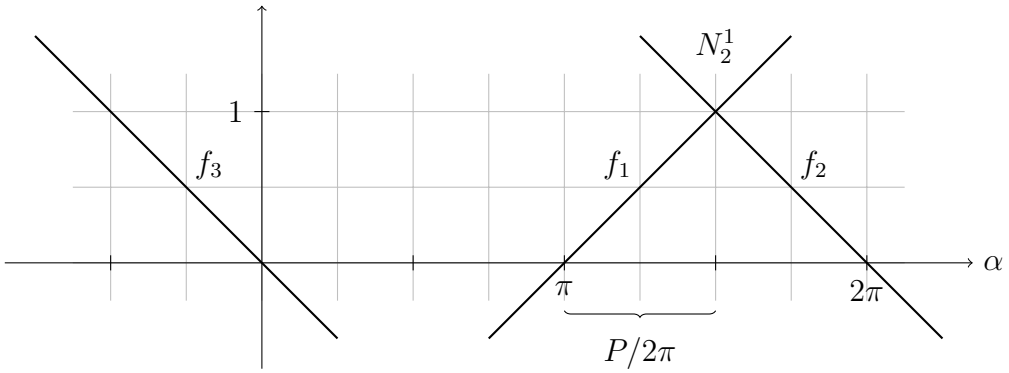


Abbildung 5.5: Beweisidee zur „kreisförmigen“ Dreiecksfunktion für B-Spline-Funktionen mit $K = 1$ und $P = 4$ als Darstellung über eine Minimum-Maximumfunktion auf drei Geraden, hier illustriert am Beispiel N_2^1.

Für $K = 1$ lässt sich weiterhin die „kreisförmige“ Dreiecksfunktion $N_p^1(\alpha)$ aus einer Reihe von Minimum- und Maximumfunktionen effizient implementieren. So gilt, dass $N_p^1(\alpha)$ zu

$$N_p^1(\alpha) = \max\Big(\min\Big(\max\Big(\frac{P}{2\pi}\alpha - p, 0\Big), \max\Big(-\frac{P}{2\pi}\alpha + p + 2, 0\Big)\Big), \max\Big(-\frac{P}{2\pi}\alpha + p + 2 - P, 0\Big)\Big)$$

vereinfacht werden kann. Dies lässt sich verifizieren, in dem die kreisförmige Dreiecksfunktion im Intervall $(0, 2\pi]$ als Verknüpfung dreier Geraden verstanden wird — zwei Geraden, die das Dreieck im Intervall $(2\pi p/P, 2\pi(p+2)/P]$ aufspannen, und einer absteigenden Geraden, die um 2π nach links verschoben wurde und dafür sorgt, die B-Spline-Funktion für $p = P - 1$ kreisförmig abzuschließen und in allen anderen Fällen keinen Anteil im Gültigkeitsbereich der Funktion $N_p^1 \colon (0, 2\pi] \to [0, 1]$ besitzt (vgl. Abbildung 5.5). Den Geraden kann die Steigung $P/2\pi$ bzw. $-P/2\pi$ zugeordnet werden und sie können damit folglich über

$$\begin{aligned} f_1(\alpha) &:= \frac{P}{2\pi}\Big(\alpha - 2\pi\frac{p}{P}\Big) = \frac{P}{2\pi}\alpha - p \\ f_2(\alpha) &:= -\frac{P}{2\pi}\Big(\alpha - 2\pi\frac{p+2}{P}\Big) = -\frac{P}{2\pi}\alpha + p + 2 \\ f_3(\alpha) &:= -\frac{P}{2\pi}\Big(\alpha - 2\pi\frac{p+2}{P} + 2\pi\Big) = -\frac{P}{2\pi}\alpha + p + 2 - P \end{aligned}$$

beschrieben werden. Mittels $\max(f_i(\alpha), 0) \in \mathbb{R}_+$ können diese auf den positiven reellen Raum eingegrenzt werden. $f_\triangle := \min(\max(f_1, 0), \max(f_2, 0))$ beschreibt da-

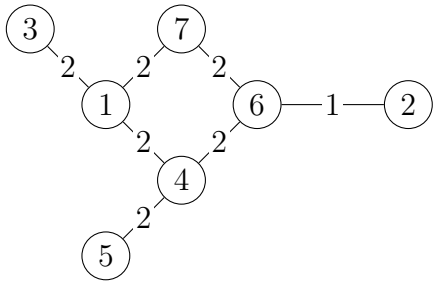
mit die Dreiecksfunktion, indem sie sich stets für das Minimum von $\max(f_1, 0)$ bzw. $\max(f_2, 0)$ entscheidet. Folglich beschreibt $\max(\max(f_3, 0), f_\triangle)$ die B-Spline-Funktion N_p^1 im Intervall $(0, 2\pi]$ für alle $p \in \{0, \ldots, P-1\}$.

5.5 Pooling auf Graphen

Neben den Faltungsschichten gehören die sogenannten Pooling-Schichten zu den fundamentalen Schichten eines CNNs. Pooling-Schichten eines Netzes, üblicherweise über Max-Pooling realisiert, werden dabei für gewöhnlich direkt nach einer Faltungsschicht benutzt und sorgen dafür, die Ausgabe der Faltung zu filtern bzw. zu reduzieren [18].
Eine Pooling-Operation auf Graphen erfordert dafür analog zum Pooling auf einem regulären Gitter eine logische Zusammenfassung von Knoten zu Clustern [5]. Der zu verwendende Clusteralgorithmus hat dabei jedoch einige Anforderungen, um als Pooling-Operation in einem Netz genutzt werden zu könnnen. So muss dieser voralem als ein mehrstufiges Clustering fungieren, bei dem jede Stufe eines Graphen den Blick auf den Graphen bei einer unterschiedlichen „Auflösung“ zeigt und insbesondere die zugrunde liegende Geometrie des Graphen erhält [5]. Das Clustering von Knoten wird daher in dieser Arbeit auch oft als *Vergröberung* eines Graphen betitelt. Die mehrfache Anwendung eines Clusteralgorithmus erlaubt damit die mehrfache Benutzung von Pooling-Schichten im Netz. Es ist weiterhin notwendig, dass die Pooling-Operation benutzerspezifische Pooling-Größen ermöglicht. Hier erscheinen vorallem Clustertechniken sinnvoll, die die Größe eines Graphen um den Faktor zwei reduzieren [5]. Damit können größere Pooling-Größen über die mehrfache Anwendung des Clusteralgorithmus realisiert werden.

5.5.1 Graphvergröberung

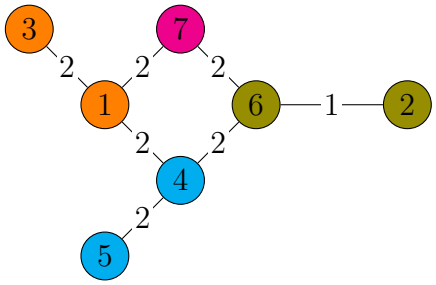
Es existieren eine Reihe von Clustertechniken auf Graphen [5, 6, 17]. Defferrard u. a. benutzen für die Pooling-Schicht eines Netzes auf Graphen die *Vergröberungsphase* des mehrstufigen Clusteralgorithmus *Graclus* [6]. Graclus zeigt sich dabei als besonders erfolgreich über einer Vielzahl von Graphen bei minimaler berechnungsintensiver Komplexität [5]. Abbildung 5.6 illustriert den Ablauf des Algorithmus. Dabei wird ein initialer Graph $\mathcal{G}_0$ sukzessive in kleinere Graphen $\mathcal{G}_1, \mathcal{G}_2, \ldots, \mathcal{G}_L$ mit $|\mathcal{V}_0| > |\mathcal{V}_1| > \cdots > |\mathcal{V}_L|$, $L \in \mathbb{N}$, transformiert [6]. Für die Transformation von einem Graphen $\mathcal{G}_l$ zu einem Graphen $\mathcal{G}_{l+1}$ mit kleinerer Knotenanzahl $|\mathcal{V}_{l+1}| < |\mathcal{V}_l|$ werden aus disjunkten Knotenuntermengen von $\mathcal{V}_l$ *Superknoten* für $\mathcal{V}_{l+1}$ gebildet [6]. Die



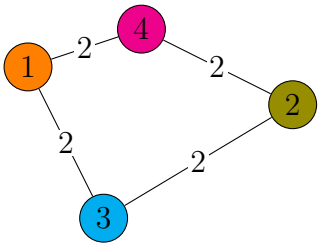
(a) Graph mit zufälliger Ordnung

Kante	Gewicht	Normalized-Cut
$(1,3)$	2	$\mathbf{1.\overline{3}}$
$(1,4)$	2	$0.\overline{3}$
$(1,7)$	2	$0.8\overline{3}$
$(2,6)$	1	$\mathbf{1.2}$
$(4,5)$	2	$\mathbf{1.\overline{3}}$
$(4,6)$	2	$0.7\overline{3}$
$(6,7)$	2	$\mathbf{0.9}$

(b) Bestimmung des Normalized-Cuts



(c) Clustering der Knoten



(d) vergröberter Graph

Abbildung 5.6: Vergröberung eines Graphen $\mathcal{G}$ mittels Graclus und Normalized-Cut bei zufälliger Knotenordung, hier angegeben über die Knotenindizes von $\mathcal{G}$ (a). Die Maxima des Normalized-Cuts (b) zwischen den Knoten und deren lokalen Nachbarschaften bestimmen in aufsteigender Reihenfolge das (höchstens) paarweise Clustering von $\mathcal{V}$, insbesondere sorgt der Normalized-Cut für die präferierte Verschmelzung mit den Blättern des Graphen (c). Der vergröberte Graph ergibt sich dann als clusterweise Aufsummierung der Kanten ohne Schleifen (d).

Kantengewichte $\mathcal{E}_{l+1}$ werden dann über die Summe der Kantengewichte der jeweiligen Knotenuntermengen ohne Schleifen gebildet [6].

Die Auswahl der Untermengen erfolgt gierig. Die Knoten des Graphen $\mathcal{G}_l$ werden als unmarkiert initialisert und zufällig durchlaufen. Für jeden Knoten $v_i \in \mathcal{V}_l$, der noch unmarkiert ist, wird ein lokaler, ebenfalls noch unmarkierter, Nachbarschaftsknoten $v_j \in \mathcal{N}(v_i)$ nach einer zuvor definierten Strategie bestimmt und v_i sowie v_j zu einem Superknoten $v^* := \{v_i, v_j\} \in \mathcal{V}_{l+1}$ verschmelzt. Anschließend werden v_i sowie v_j markiert. Falls v_i keinen unmarkierten Nachbarsknoten besitzt, wird v_i alleine als Superknoten $v^* := \{v_i\} \in \mathcal{V}_{l+1}$ deklariert und markiert. Der Algorithmus ist abgeschlossen, sobald alle Knoten erfolgreich markiert wurden [6]. Bei weiteren Anwendungen des Clusteralgorithmus werden die Knoten im Folgenden anhand ihrer

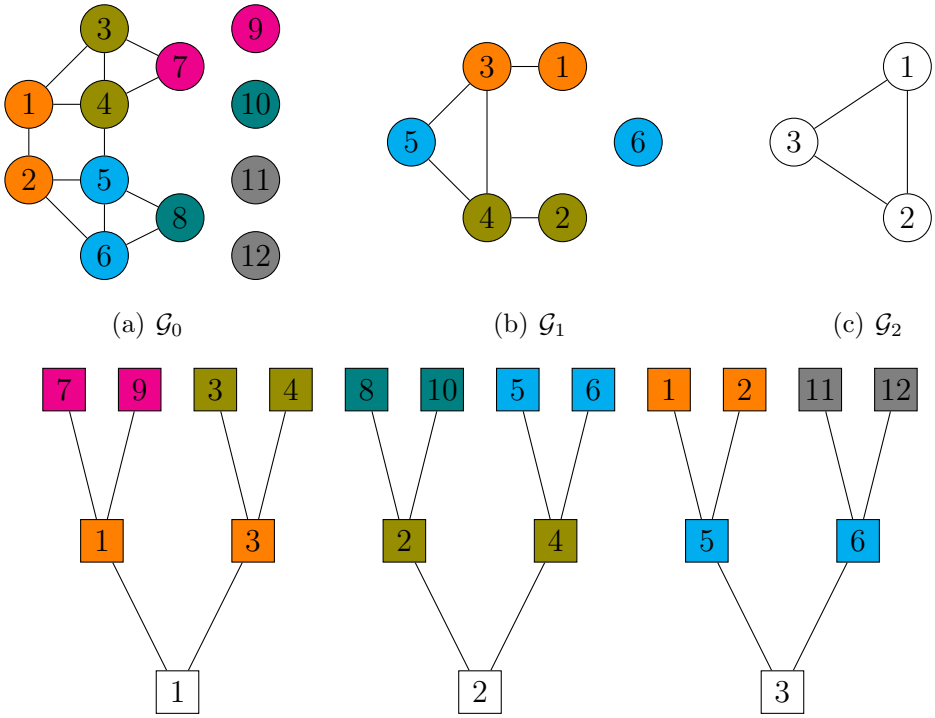
aufsteigenden Knotengrade durchlaufen [5].
Strategien für die Nachbarschaftsauswahl basieren üblicherweise auf der Maximierung von $w(v_i, v_j)$ [6]. Eine darauf aufbauende Strategie ist der *Normalized-Cut*

$$\mathrm{NCut}(v_i, v_j) \coloneqq w(v_i, v_j)\left(\frac{1}{d(v_i)} + \frac{1}{d(v_j)}\right),$$

der die Kantengewichte relativ zu ihrem Knotengrad gewichtet [5, 6]. Der Normalized-Cut sorgt dafür, dass Kanten als wichtiger gezählt werden, wenn ihre entsprechenden Knoten einen geringen Grad besitzen und damit folglich als unwahrscheinlicher gelten, mit einem anderen Knoten zu verschmelzen.
Graclus reduziert die Knotenanzahl eines beliebigen Graphen näherungsweise um die Hälfte, d.h. $2|\mathcal{V}_{l+1}| \approx |\mathcal{V}_l|$. Es können jedoch vereinzelt Superknoten entstehen, die nur aus einem einzigen Knoten der vorangegangenen Schicht gebildet wurden. In der Praxis zeigt sich jedoch, dass Graclus nur sehr wenige solcher Superknoten generiert [5]. Es ist weiterhin wichtig anzumerken, dass der Algorithmus bei mehrmaliger Anwendung auf dem gleichen Graphen aufgrund seiner zufälligen Iteration auf den Knoten zwei verschiedene, vergröberte Graphen erzeugen kann. Damit kann der Prozess des Clusterings bereits als ein Augmentierungsschritt der Eingabedaten verstanden werden.

5.5.2 Effizientes Pooling mittels binärer Bäumen

Ein tiefes neuronales Netz besitzt für gewöhnlich viele Pooling-Schichten. Eine Pooling-Operation auf den Merkmalen der Graphknoten muss folglich vorallem effizient sein. Ein naiver Ansatz dafür ist eine „Lookup“-Tabelle, in der die Verbindungen der Knoten zu ihren Superknoten gespeichert sind. Eine Pooling-Operation muss demnach für jedes Cluster ihre Knoten in der Tabelle referenzieren und ihre Operation auf diesen vollführen. Das resultiert in einer extrem speicherineffizienten, langsamen und schwer zu parallelisierenden Implementierung [5]. Es ist jedoch möglich, die Knoten des Graphen so anzuordnen, dass eine Pooling-Operation auf diesen in linearer Zeit, d.h. $\mathcal{O}(N)$, realisiert werden kann [5].
Nach jedem Vergröberungsschritt besitzt ein Knoten entweder genau einen oder zwei Kinder, jenachdem ob es zum Zeitpunkt der Betrachtung noch einen unmarkierten Nachbarn gibt. Falls ein Knoten v im Graphen $\mathcal{G}_{l+1}$ nur ein Kind besitzt, so wird ein „Fake“-Knoten ohne Kanten in den Graphen $\mathcal{G}_l$ hinzugefügt [5]. Damit besitzt jeder Knoten folglich nun immer zwei Kinder. Foglich kann über die Eltern-Kind-Beziehung von der L-ten bis zur 0-ten Stufe ein balancierter, binärer Baum aufgebaut



(a) $\mathcal{G}_0$ (b) $\mathcal{G}_1$ (c) $\mathcal{G}_2$

(d) Anorderung der Knoten zu einem binären Baum

Abbildung 5.7: Illustration einer Pooling-Operationen der Größe 4 (bzw. zweier Pooling-Operationen der Größe 2) auf einem Graphen $\mathcal{G}$ der Größe $|\mathcal{V}| = 8$. Das Clustering von $\mathcal{G}$ liefert uns im ersten Schritt $N_1 = |\mathcal{V}_1| = 5$ Knoten und im darauf folgenden $N_2 = |\mathcal{V}_2| = 3$ Knoten. Die Größen der Graphen werden daraufhin zu $N_2 = 3$, $N_1 = 6$ und $N_0 = 12$ angepasst, so dass jeder Knoten auf genau 2 Vorgängerknoten verweist, indem „Fake"-Knoten zu $\mathcal{G}_1$ (1 Knoten) und $\mathcal{G}_0$ (4 Knoten) hinzugefügt werden. Mit der Anordnung der Knoten zu einem binären Baum (d) kann die Pooling-Operation $\max(\cdot)$ eines Signals $\mathbf{f} \in \mathbb{R}^{12}$ auf $\mathcal{G}_0$ dann effizient als $\mathbf{f}_{\text{pool}} := [\max(\mathbf{f}_1, \mathbf{f}_2, \mathbf{f}_3, \mathbf{f}_4), \max(\mathbf{f}_5, \mathbf{f}_6, \mathbf{f}_7, \mathbf{f}_8), \max(\mathbf{f}_9, \mathbf{f}_{10}, \mathbf{f}_{11}, \mathbf{f}_{12})]^\top \in \mathbb{R}^3$ implementiert werden, wobei die Werte der „Fake"-Knoten $\mathbf{f}_2, \mathbf{f}_6, \mathbf{f}_{11}, \mathbf{f}_{12}$ auf den neutralen Wert Null gesetzt werden.

werden. Reguläre Knoten aus dem Graphen besitzen damit entweder (a) genau zwei reguläre Knoten, (b) einen regulären Knoten und einen „Fake"-Knoten als Kinder und (c) „Fake-Knoten" besitzen immer genau zwei „Fake"-Knoten als Kinder [5]. Abbildung 5.7 illustriert die Konstruktion eines solchen Baumes. Die Merkmale an den „Fake"-Knoten eines Graphen werden mit einem neutralen Wert initialisiert, d.h. zum Beispiel mit Null bei einem Netz mit ReLU-Aktivierungsfunktion und der Verwendung von Max-Pooling als Pooling-Operation [5]. Ebenso kann auf diesen „Fake"-Knoten weiterhin ohne Einschränkungen gefaltet werden, da sie keinerlei Nachbarsknoten besitzen. Die Anordnung der Knoten zu einem balancierten, binären

Baum liefert uns eine Ordnung auf den Knoten von $\mathcal{G}_0$ bis $\mathcal{G}_{L-1}$, auf der wir eine Pooling-Operation völlig analog zu einem eindimensionalen Gitter mit einer Größe und Schrittweise von zwei definieren können [5]. Diese Anordnung der Knoten macht den Prozess des Poolings extrem effizient und erlaubt ebenso eine parallele Verarbeitungsarchikektur auf GPUs [5].

Größere Pooling-Größen müssen ein Vielfaches von zwei sein und können dann ebenso leicht über eine tieferstufigere Vergröberung implementiert werden (vgl. Abbildung 5.7).

5.5.3 Erweiterung auf Graphen im zweidimensionalen Raum

Sei $\mathcal{G}_l = (\mathcal{V}_l, \mathcal{E}_l, \mathrm{p}_l)$ ein Graph im zweidimensionalen euklidschen Raum, wobei die Position der Knoten des Graphen eindeutig über die Funktion $\mathrm{p}_l \colon \mathcal{V}_l \to \mathbb{R}^2$ bestimmt ist (vgl. Kapitel 3). Dann lässt sich der mehrstufige Clusteralgorithmus Graclus aus Kapitel 5.5.1 insofern anpassen, dass dieser die Vergröberung eines Graphen auch in Bezug auf die Position seiner Superknoten in der Ebene wiederspiegelt. Sei dafür weiterhin $v^* \coloneqq \{v_i, v_j\} \in \mathcal{V}_{l+1}$, ein Superknoten der Knoten $v_i, v_j \in \mathcal{V}_l$. Dann ergibt sich die neue Position von v^* als

$$\mathrm{p}_{l+1}(v^*) \coloneqq \frac{\mathrm{p}_l(v_i) + \mathrm{p}_l(v_j)}{2}.$$

Für Superknoten $v^* \coloneqq \{v_i\} \in \mathcal{V}_{l+1}$ mit einem einzigen Knoten $v_i \in \mathcal{V}_l$ bleibt die Position unverändert mit $\mathrm{p}_{l+1}(v^*) \coloneqq \mathrm{p}_l(v_i)$. Die Adjazenzmatrizen $\mathbf{A}_{\mathrm{dist}} \in [0,1)^{N \times N}$ und $\mathbf{A}_{\mathrm{rad}} \in [0, 2\pi]^{N \times N}$ zu $\mathcal{G}_{l+1}$ können dann analog zu Kapitel 3 aus $\mathcal{V}_{l+1}$, $\mathcal{E}_{l+1}$ und p_{l+1} gewonnen werden.

Wohnt den Knoten in $\mathcal{G}_l = (\mathcal{V}_l, \mathcal{E}_l, \mathrm{p}_l, \mathrm{m}_l)$ wie in Kapitel 3.2 zusätzlich eine Masse $\mathrm{m}_l \colon \mathcal{V}_l \to \mathbb{R}$ bei, so gewichtet sich die Position des Superknotens $v^* \in \mathcal{V}_{l+1}$ abhängig der Massen $\mathrm{m}(v_i)$ und $\mathrm{m}(v_j)$ als

$$\mathrm{p}_{l+1}(v^*) \coloneqq \frac{\mathrm{m}_l(v_i)\mathrm{p}_l(v_i) + \mathrm{m}_l(v_j)\mathrm{p}_l(v_j)}{\mathrm{m}_l(v_i) + \mathrm{m}_l(v_j)}.$$

Die neue Masse des Superknotens ergibt sich dann als Vereinigung der Massen $\mathrm{m}_{l+1}(v^*) \coloneqq \mathrm{m}_l(v_i) + \mathrm{m}_l(v_j)$. Damit erhält der vergröberte Graph $\mathcal{G}_{l+1}$ die Positionen und Massen seiner Vorgängerknoten des Graphen $\mathcal{G}_l$ im zweidimensionalen euklidschen Raum.

5.6 Netzarchitektur

6 Evaluation

6.1 Versuchsaufbau

6.1.1 Datensätze

MNIST. [16]

Cifar-10. [15]

Pascal VOC. [8]

6.1.2 Metriken

6.1.3 Parameterwahl

Vorstellung aller Parameter Superpixelalgorithmen Parameterwahl

Augmentierung von Graphen.

6.2 Merkmalsselektion

6.3 Ergebnisse

Vergleich mit anderen Implementierungen. Alle Faltungen wurden dabei mit einer Partitionsgröße von 8 bei $K = 0$ und $K = 1$ implementiert, um ein CNN mit einem 3×3 Filter zu simulieren. Es erscheint jedoch vorstellbar die Filtergröße bei größerer lokaler Kontrollierbarkeit, d.h. $K > 1$, weiter zu reduzieren und die Gefahr des Overfittings damit auf Grund der kleineren Anzahl an Trainingsparametern einzuschränken.

6.4 Laufzeitanalyse

Vergleich mit anderen Implementierungen.

6.5 Diskussion

7 Ausblick

Weitere Anwendungsgebiete.

Entfernung irrelevanter Knoten. nicht nur bei MNIST auch bei pascal voc slic? (zum Beispiel Regelmäßigkeiten erkennen)

Spatial-Pyramid-Pooling.

Attention-Algorithmus.

A Weitere Informationen

Abbildungsverzeichnis

5.1 5-Punkte-Stern 14
5.2 Graphrepräsentation eines regulären Gitters 24
5.3 Partitionierung eines Graphknotens 25
5.4 Geschlossene B-Spline-Funktionen 29
5.5 B-Spline-Funktion mit $K = 1$ über Minimum-Maximumfunktion . . . 30
5.6 Graphvergröberung mittels Graclus und Normalized-Cut 32
5.7 Pooling auf Graphen 34

Algorithmenverzeichnis

5.1 Weisfeiler-Lehman Algorithmus 22

Literaturverzeichnis

[1] Abramowski, Stephan; Müller, Heinrich: *Geometrisches Modellieren.* B.I. Wissenschaftsverlag, 1991 (Reihe Informatik)

[2] Achanta, Radhakrishna; Shaji, Appu; Smith, Kevin; Lucchi, Aurelien; Fua, Pascal; Susstrunk, Sabine: SLIC Superpixels Compared to State-of-the-Art Superpixel Methods. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2012), S. 2274–2282

[3] Chung, Fan .R.K.: *Spectral Graph Theory.* American Mathematical Society, 1997

[4] de Boor, Carl: *A Practical Guide to Splines.* Springer Verlag, 1978

[5] Defferrard, Michaël; Bresson, Xavier; Vandergheynst, Pierre: Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering. In: *Advances in Neural Information Processing Systems 29.* Curran Associates, Inc., 2016, S. 3844–3852

[6] Dhillon, Inderjit S.; Guan, Yuqiang; Kulis, Brian: Weighted Graph Cuts Without Eigenvectors: A Multilvel Approach. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2007), S. 1944–1957

[7] Douglas, Brendan L.: The Weisfeiler-Lehman Method and Graph Isomorphism Testing. In: *arXiv preprint arXiv:1101.5211* (2011)

[8] Everingham, Mark; Eslami, S.M. Ali; Van Gool, Luc; Williams, Christoper K. I.; Winn, John; Zisserman, Andrew: The Pascal Visual Object Classes Challenge: A Retrospective. In: *International Journal of Computer Vision* (2015), S. 98–136

[9] Felzenszwalb, Pedro F.; Huttenlocher, Daniel P.: Efficient Graph-Based Image Segmentation. In: *International Journal of Computer Vision* (2004), S. 167–181

[10] Hammond, David K.; Vandergheynst, Pierre; Gribonval, Réne: Wavelets on Graphs via Spectral Graph Theory. In: *Applied and Computational Harmonic Analysis* (2011), S. 129–150

[11] He, Kaiming; Zhang, Xiangyu; Ren, Shaoqing; Sun, Jian: Deep Residual Learning for Image Recognition. In: *Computer Vision and Pattern Recognition* (2016), S. 83–98

[12] Hoffman, Kenneth; Kunze, Ray Alden: *Linear Algebra*. Prentice-Hall, 1971

[13] Huszár, Ferenc: *How Powerful are Graph Convolutions?* `http://www.inference.vc/how-powerful-are-graph-convolutions-review-of-kipf-welling-2016-2/`. 2016

[14] Kipf, Thomas N.; Welling, Max: Semi-Supervised Classification with Graph Convolutional Networks. In: *Computing Research Repository* (2016)

[15] Krizhevsky, Alex: *Learning Multiple Layers of Features from Tiny Images*, Department of Computer Science, University of Toronto, Diplomarbeit, 2009

[16] LeCun, Yann; Cortes, Corinna; Burges, Christopher J.C.: The MNIST Database of Handwritten Digits. (2010)

[17] Luxburg, Ulrike: A Tutorial on Spectral Clustering. In: *Statistics and Computing* (2007), S. 395–416

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

[19] Reuter, Martin; Biasotti, Silvia; Giorgi, Daniela; Patanè, Guiseppe; Spagnuolo, Michela: Discrete Laplace-Beltrami Operators for Shape Analysis and Segmentation. In: *Computers & Graphics* (2009), S. 381–390

[20] Shuman, David I.; Narang, Sunil. K.; Frossard, Pascal; Ortega, Antonio; Vandergheynst, Pierre: The Emerging Field of Signal Processing on Graphs: Extending High-Dimensional Data Analysis to Networks and Other Irregular Domains. In: *IEEE Signal Processing Magazine* (2013), S. 83–98

[21] Simonyan, Karen; Zisserman, Andrew: Very Deep Convolutional Networks for Large-Scale Image Recognition. In: *Computing Research Repository* (2014)

[22] Vedaldi, Andrea; Soatto, Stefano: Quick Shift and Kernel Methods for Mode Seeking. In: *European Conference on Computer Vision*, 2008, S. 705–718

[23] Weisfeiler, Boris; Lehman, A. A.: A Reduction of a Graph to a Canonical Form and an Algebra Arising During this Reduction. In: *Nauchno-Technicheskaya Informatsia* (1968), S. 12–16

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