technische universität
dortmund

fakultät für informatik

Master-Thesis

Convolutional Neural Networks auf Graphrepräsentationen von Bildern

Matthias Fey

25. Juni 2017

Gutachter:

Prof. Dr. Heinrich Müller

M.Sc. Jan Eric Lenssen

Lehrstuhl Informatik VII
Graphische Systeme
TU Dortmund

Inhaltsverzeichnis

1 Einleitung

Homepage[1]

1.1 Problemstellung

1.2 Aufbau der Arbeit

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

2 Grundlagen

Das folgende Kapitel erläutert die Grundlagen und Notationen, die zum weiteren Verständnis der Arbeit benötigt werden. Zunächst werden in Unterkapitel 2.1 die grundlegenden Notationen zu Mengen, Vektoren, Matrizen und Tensoren definiert, die im weiteren Verlauf verwendet werden. Daraufhin folgt in Unterkapitel 2.2 eine kurze Einführung in das Gebiet der Graphentheorie. Abschließend führt das Unterkapitel 2.3 in das Gebiet der neuronalen Netze und insbesondere der Convolutional Neural Networks ein, auf denen diese Arbeit aufbaut. Für einen umfassenderen Blick in das Gebiet des *Deep Learnings*, den diese Arbeit nicht leisten kann, sei auf Nielsen [24] verwiesen.

2.1 Mathematische Notationen

Eine *ungeordnete Menge* $\mathcal{A} \coloneqq \{a_1, \ldots, a_N\} = \{a\}_{n=1}^{N}$, $N \in \mathbb{N}$, beschreibt eine Gruppierung von N Elementen $a_n \in \mathcal{A}$. Eine Menge $\mathcal{A}$ heißt *geordnet*, wenn auf dessen Elementen eine reflexive, transitive und antisymmetrische Ordnung $\mathcal{A} \times \mathcal{A}$ definiert ist [17]. Eine geordnete Menge $\mathcal{A}$ mit N Elementen wird über $\mathcal{A} \coloneqq (a_1, \ldots, a_N)$ gekennzeichnet, wobei die Reihenfolge der Elemente $(a_1, \ldots, a_N)$ dessen Ordnung beschreiben.

Sei $\mathbf{v} \in \mathbb{R}^N$, $N \in \mathbb{N}$, mit $\mathbf{v} = [v_1, \ldots, v_n]^\top$ ein *Vektor* mit N reellen Elementen $\{v_n\}_{n=1}^{N}$, $v_n \in \mathbb{R}$, wobei $\mathbf{v}_n = v_n$ das n-te Element von $\mathbf{v}$ referenziert. Zu zwei Vektoren $\mathbf{v}, \mathbf{w} \in \mathbb{R}^N$ ist das *Skalarprodukt* $\langle \mathbf{v}, \mathbf{w} \rangle$ definiert als $\langle \mathbf{v}, \mathbf{w} \rangle \coloneqq \sum_{n=1}^{N} \mathbf{v}_n \mathbf{w}_n$ [17]. $\mathbf{v}$ und $\mathbf{w}$ stehen *orthogonal* zueinander, d.h. $\mathbf{v} \perp \mathbf{w}$, genau dann, wenn $\langle \mathbf{v}, \mathbf{w} \rangle = 0$ [17].

Eine *Matrix* $\mathbf{M} \in \mathbb{R}^{N \times M}$, $N, M \in \mathbb{N}$, erweitert einen Vektor um M Spalten. Der Wert $\mathbf{M}_{nm} \in \mathbb{R}$ beschreibt damit das Element in der n-ten Zeile und m-ten Spalte von $\mathbf{M}$. Die *transponierte Matrix* $\mathbf{M}^\top \in \mathbb{R}^{M \times N}$ ist eine Matrix, die aus $\mathbf{M} \in \mathbb{R}^{N \times N}$ durch Vertauschen der Zeilen und Spalten entsteht, d.h. $\mathbf{M}^\top_{mn} = \mathbf{M}_{nm}$ [17]. Zu einem Vektor $\mathbf{v} \in \mathbb{R}^N$ lässt sich weiterhin dessen korrespondiere quadratische *Diagonalma-*

trix $\operatorname{diag}(\mathbf{v}) \in \mathbb{R}^{N \times N}$ über

$$\operatorname{diag}(\mathbf{v}) \coloneqq \begin{bmatrix} \mathbf{v}_1 & & \mathbf{0} \\ & \ddots & \\ \mathbf{0} & & \mathbf{v}_N \end{bmatrix}$$

bzw. $\operatorname{diag}(\mathbf{v})_{nn} \coloneqq \mathbf{v}_n$ definieren [6].

Dünnbesetzte Matrizen. Eine *dünnbesetzte* oder *schwachbesetzte Matrix* ist eine Matrix, bei der so viele Einträge aus Nullen bestehen, dass sich statt der üblichen Speicherung einer Matrix als zweidimensionales Feld speichereffizientiere Datenstrukturen ergeben. In der Regel gilt eine Matrix $\mathbf{M} \in \mathbb{R}^{N \times N}$ als dünnbesetzt, wenn diese nicht aus mehr als N oder $N \log N$ Einträgen ungleich Null besteht. Neben dem Speichergewinn lassen sich viele Operationen auf Matrizen ebenso berechnungseffizienter implementieren [27]. So muss zum Beispiel zur Bestimmung des größten Elements einer Matrix nicht die komplette Matrix, sondern lediglich deren explizit eingetragene Werte betrachtet werden. Es gibt jedoch auch Operationen, die nicht auf dünnbesetzten Matritzen definiert sind, sodass diese vorher in eine *dichte* Matrix überführt werden müssen (vgl. [27]). Im Laufe dieser Arbeit haben wir es oft mit dünnbesetzten Matrizen zu tun. So wird zum Beispiel eine Diagonalmatrix *immer* als eine dünnbesetzte Matrix implementiert.

Tensoren. Ein *Tensor* $\mathbf{T} \in \mathbb{R}^{N_1 \times \cdots \times N_R}$, $N_1, \ldots, N_R \in \mathbb{N}$, ist ein mathematisches Objekt, der das Konzept der Vektoren und Matrizen auf beliebig viele Dimensionen erweitert. Die Anzahl der Dimensionen R eines Tensors wird auch *Rang* genannt [17]. Vektoren können damit insbesondere als Tensor mit Rang 1 sowie Matrizen als Tensor mit Rang 2 verstanden werden. Ein Tensor mit Rang 0 beschreibt ein Skalar. Aus einem Tensor $\mathbf{T} \in \mathbb{R}^{N_1 \times \cdots \times N_R}$ können über die Indexnotation Tensoren mit geringerer Dimension gefiltert werden. So beschreibt $\mathbf{T}_{n_1 \cdots n_r}$ einen Tensor mit Rang 0 bzw. das Element des Tensors an Position $n_1, \ldots, n_r$. Analog beschreibt zum Beispiel $\mathbf{T}_{n_1} \in \mathbb{R}^{N_2 \times \cdots \times N_R}$ einen Tensor mit Rang $R - 1$, bei dem die erste Dimension über n_1 festgehalten wird.

Bilder. Ein *Bild* kann folglich durch einen dreidimensionalen Tensor $\mathbf{B} \in \mathbb{R}^{H \times W \times C}$ repräsentiert werden, wobei $H, W \in \mathbb{N}$ die Höhe bzw. Breite des Bildes angeben und $C \in \{1, 3\}$ die Anzahl der Farbkanäle des Bildes beschreibt, d.h. ein Graubild mit nur einem Kanal oder ein Farbbild mit drei Kanälen (zum Beispiel über das

RGB- oder Lab-Farbmodell). Ein *Pixel* eines Bildes $\mathbf{B}$ an der Position (x, y) mit $1 \leq x \leq W, 1 \leq y \leq H$ kann folglich über $\mathbf{B}_{yx} \in \mathbb{R}^C$ angesprochen werden.

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 $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ [3]. 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 [3]. 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 [28].

Ein *gewichteter Graph* $\mathcal{G} = (\mathcal{V}, \mathcal{E}, w)$ ist ein Graph, der zusätzlich eine *Gewichtsfunktion* $w\colon \mathcal{V} \times \mathcal{V} \to \mathbb{R}_+$ auf den Kanten des Graphen definiert, sodass $(v_i, v_j) \notin \mathcal{E}$ genau dann, wenn $w(v_i, v_j) = 0$ [3]. Im Falle eines ungewichteten Graphen ist die Gewichtsfunktion w implizit durch $\mathcal{E}$ über $w\colon \mathcal{V} \times \mathcal{V} \to \{0, 1\}$ gegeben.

Ein Graph heißt *ungerichtet*, falls $w(v_i, v_j) = w(v_j, v_i)$ [3]. 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 [3]. 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 [6]. 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 [6]. Die unterschiedliche Notation macht deutlich, welcher Grad eines Knotens gemeint ist. 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. $\mathbf{D}_{ii} = d(v_i)$ [6]. Ein Knoten $v \in \mathcal{V}$ heißt *isoliert*, falls dieser keinen Nachbarsknoten besitzt, d.h. $\deg(v) = 0$ [6].

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 den Indizes der Knoten $\{v_n\}_{n=1}^N$ [3]. 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 [13]. v_j ist von v_i

aus *erreichbar*, wenn $s(v_i, v_j) \in \mathbb{N}$ [3]. Die Relation der Erreichbarkeit in der Knotenmenge $\mathcal{V}$ eines Graphen ist eine Äquivalenzrelation. Die Äquivalenzklassen der Erreichbarkeitsrelation heißen die *Zusammenhangskomponenten* des Graphen $\mathcal{G}$ [3]. Wir nennen $\mathcal{G}$ *zusammenhängend*, wenn $\mathcal{G}$ genau eine Zusammenhangskomponente besitzt, d.h. zu jedem Knoten v_i existiert mindestens ein Weg zu jedem anderen Knoten $v_j \in \mathcal{V}$ [13]. 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 [13].
Eine Funktion $f \colon \mathcal{V} \to \mathbb{R}^M$ auf den Knoten eines Graphen $\mathcal{G}$, die auf einen M-dimensionalen Vektor abbildet, heißt *Merkmalsfunktion* der Knotenmenge. Eine Merkmalsfunktion f kann ebenso als *Merkmalsmatrix* $\mathbf{F} \in \mathbb{R}^{N \times M}$ mit $\mathbf{F}_{im} \coloneqq f(v_i)_m$ interpretiert werden [28]. Bildet f lediglich auf ein einziges Element ab, d.h. $M = 1$ und folglich $f \colon \mathcal{V} \to \mathbb{R}$, ergibt sich daraus analog ein *Merkmalsvektor* $\mathbf{f} \in \mathbb{R}^N$ mit $\mathbf{f}_i \coloneqq f(v_i)$.

2.3 Convolutional Neural Networks

conv2d Convolutional Neural Network (CNN)

3 Graphrepräsentationen von Bildern

Als eine *Graphrepräsentation eines Bildes* $\mathbf{B} \in \mathbb{R}^{H \times W \times C}$ wird eine Darstellung von $\mathbf{B}$ als ein gewichteter, ungerichteter sowie schleifenloser Graph $\mathcal{G}$ verstanden, deren Knoten Informationen zu ausgewählten Bereichen von $\mathbf{B}$ über eine Merkmalsmatrix $\mathbf{F} \in \mathbb{R}^{N \times M}$ speichern und deren Kanten eine Aussage über die örtlichen Nachbarschaften eines jeden Bildbereichs inne wohnt. Formal lässt sich eine Graphrepräsentation eines Bildes damit als ein *Graph im zweidimensionalen euklidischen Raum* $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathrm{p})$ verstehen, dem zusätzlich zu seinen Knoten- und Kantenmengen anstatt einer Gewichtsfunktion $w \colon \mathcal{V} \times \mathcal{V} \to \mathbb{R}$ eine Positionsfunktion $\mathrm{p} \colon \mathcal{V} \to \mathbb{R}^2$ auf seinen Knoten in den zweidimensionalen euklidischen Raum $\mathbb{R}^2$ zugeordnet ist. Das Gewicht $w \colon \mathcal{E} \to [0, 1]$ einer Kante ergibt sich dann implizit als „Abstandsfunktion“ mit Hilfe der euklidischen Norm $\|\cdot\|_2$ auf den Positionen des Knotens über

$$\|\mathrm{p}(v_i) - \mathrm{p}(v_j)\|_2 \coloneqq \sqrt{\Big(\mathrm{p}(v_i)_1 - \mathrm{p}(v_j)_1\Big)^2 + \Big(\mathrm{p}(v_i)_2 - \mathrm{p}(v_j)_2\Big)^2}$$

und der Gaußfunktion als

$$w(v_i, v_j) \coloneqq \begin{cases} \exp\Big(-\frac{\|\mathrm{p}(v_i) - \mathrm{p}(v_j)\|_2^2}{2\xi^2}\Big), & \text{wenn } (v_i, v_j) \in \mathcal{E}, \\ 0, & \text{sonst} \end{cases} \tag{3.1}$$

mit der Standardabweichung $\xi \in \mathbb{R}$ [28]. Abbildung 3.1 veranschaulicht die Gewichtsfunktion anhand unterschiedlich gewählter ξ. Aufgrund der Symmetrie von $\|\cdot\|_2$ folgt insbesondere sofort die Ungerichtheit des Graphen $\mathcal{G}$, d.h. $w(v_i, v_j) = w(v_j, v_i)$. Die Gaußfunktion „invertiert“ dabei den Abstand zweier Knoten zueinander, sodass Knoten die weiter voneinander entfernt liegen ein geringeres Gewicht besitzen. Das korrespondiert mit dem üblichen Verständnis eines Kantengewichts in Graphen. Knoten, die näher am Ursprungsknoten liegen, gelten als „wichtiger“ und bekommen deshalb ein größeres Gewicht. Insbesondere stimmt dies mit der Konvention überein, dass $w(v_i, v_j) = 0$ für $(v_i, v_j) \notin \mathcal{E}$. Damit lässt sich weiterhin die korrespondierende Adjazenzmatrix $\mathbf{A}_{\text{dist}} \in [0, 1] \in \mathbb{R}^{N \times N}$ wie bekannt mit $(\mathbf{A}_{\text{dist}})_{ij} \coloneqq w(v_i, v_j)$ definieren. Zusätzlich zu dem Abstand der Knoten bzw. der Länge der Kanten eines Graphen im

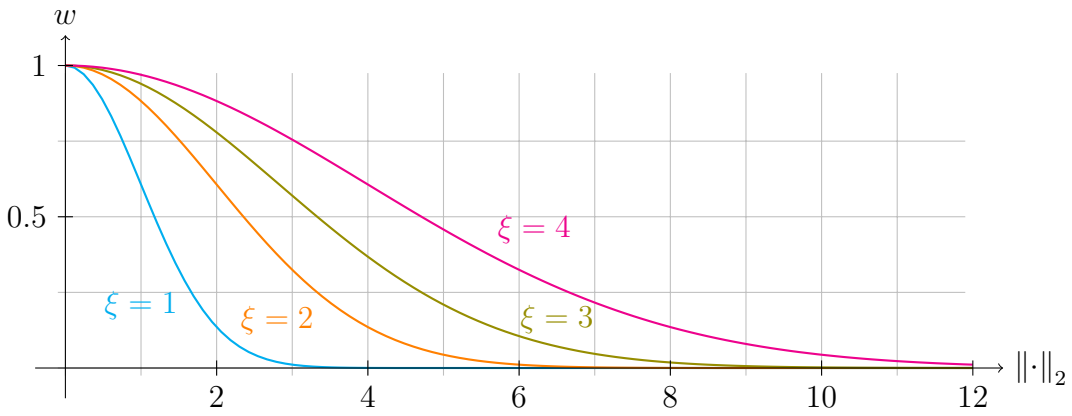


Abbildung 3.1: Illustration der „invertierten“ Kantengewichte $w \in [0,1]$ im Verhältnis zu deren Länge gegeben durch die euklidische Norm $\|\cdot\|_2$. Je größer ξ gewählt wird, umso „stumpfer“ zeichnet sich die Gaußfunktion und umso längere Kanten erhalten ein Gewicht $\gg 0$. Die Wahl von ξ ist damit abhängig von der Ausdehnung der Distanzen eines Graphen.

zweidimensionalen euklidischen Raum $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathrm{p})$ lassen sich die Ausrichtungen der Kanten über die Positionen der Knoten von $\mathcal{G}$ festhalten. Aus $p\colon \mathcal{V} \to \mathbb{R}$ wird folglich über

$$\varphi(v_i, v_j) \coloneqq \begin{cases} \operatorname{atan2}\big(\mathrm{p}(v_j - v_i)_1, \mathrm{p}(v_j - v_i)_2\big) + \pi, & \text{wenn } (v_i, v_j) \in \mathcal{E}, \\ 0, & \text{sonst} \end{cases}$$

eine *Winkelfunktion* $\varphi\colon \mathcal{V} \times \mathcal{V} \to [0, 2\pi]$ auf den Kanten von $\mathcal{G}$ im Uhrzeigersinn definiert, wobei $\operatorname{atan2}(x, y) \in [-\pi, \pi]$ den Arkustangens von x/y berechnet, aber im Gegensatz zu $\operatorname{atan}(\cdot)$ die Vorzeichen beider Parameter beachtet und so den Quadranten des Ergebnisses bestimmen kann (vgl. [25]). Der Winkel Null wird dabei explizit als 2π definiert, sodass weiterhin $\varphi(v_i, v_j) = 0$ gilt, falls $(v_i, v_j) \notin \mathcal{E}$. Analog zu $\mathbf{A}_{\text{dist}}$ lässt sich damit die Adjazenzmatrix $\mathbf{A}_{\text{rad}} \in [0, 2\pi]^{N \times N}$ mit $(\mathbf{A}_{\text{rad}})_{ij} \coloneqq \varphi(v_i, v_j)$ definieren. Es ist anzumerken, dass die Winkelfunktion φ im Gegensatz zur Gewichtsfunktion w nicht symmetrisch bzw. ungerichtet ist, d.h. $\varphi(v_i, v_i) \neq \varphi(v_j, v_i)$. Insbesondere definiert $\mathbf{A}_{\text{rad}}$ damit einen gerichteten Graphen.

Die beiden Adjazenzmatrizen $\mathbf{A}_{\text{dist}}$ und $\mathbf{A}_{\text{rad}}$ beschreiben den Graphen $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathrm{p})$ bis auf Translation eindeutig. Eine Veränderung einer Position $\mathrm{p}(v)$ eines Knotens $v \in \mathcal{V}$ führt sowohl zu einer Veränderung in $\mathbf{A}_{\text{dist}}$ als auch in $\mathbf{A}_{\text{rad}}$. Eine Translation aller Knoten um $\delta \in \mathbb{R}^2$, d.h. $\mathrm{p}(v) \to \mathrm{p}(v) + \delta$, generiert hingegen vollkommen äquivalente Adjazenzmatrizen $\mathbf{A}_{\text{dist}}$ sowie $\mathbf{A}_{\text{rad}}$. In der Regel ist dies kein Nachteil, denn nur in den seltensten Fällen interessieren uns die absoluten Positionen der

Knoten in $\mathcal{G}$, wohingegen der Abstand und die Ausrichtungen der Knoten zueinander eine größere Bedeutung genießen.
Im Folgenden werden basierend auf der Definition eines Graphen im zweidimensionalen euklidischen Raum zwei Ansätze für eine Graphrepräsentation von Bildern vorgestellt — der Repräsentation über ein reguläres Gitter sowie der Repräsentation über die Bestimmung von Superpixeln eines Bildes.

3.1 Gitter

Die einfachste Form einer Graphrepräsentation $\mathcal{G}$ eines Bildes $\mathbf{B} \in \mathbb{R}^{H \times W \times C}$ ist die Repräsentation des Bildes über einen regulären Gittergraphen, denn dafür müssen keine Berechnungen am Bild vorgenommen werden. Ein *regulärer Gittergraph im zweidimensionalen euklidischen Raum* ist ein Graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathrm{p})$, der aus einem regulären Gitter gewonnen wurde und demnach genau $N \coloneqq H \times W$ Knoten enthält, d.h. einen Knoten für jedes Pixel in $\mathbf{B}$ [6]. Die Positionsfunktion der Knoten $\mathrm{p}\colon \mathcal{V} \to \mathbb{R}^2$ entspricht damit genau der Koordinate des korrespondierenden Pixels im Urpsprungsbild. Sei dafür $v \in \mathcal{V}$ der Knoten zu dem Bildpunkt an Position (x, y). Folglich gilt für die Position des Knotens $\mathrm{p}(v) \coloneqq [x, y]^\top$. Die örtlich um den Knoten $v \in \mathcal{V}$ liegenden Knoten gelten dann auch im Gittergraph als benachbart und werden über eine Kante in $\mathcal{E}$ verbunden. Dabei unterscheidet man zwischen zwei frei wählbaren *Konnektivitäten* des Graphen [6]. Eine Konnektivität von 4 bedeutet, dass ein Knoten (ohne Berücksichtigung der Randknoten) genau vier Nachbarn besitzt, die horizontal und vertikal zu im stehen. Bei einer Konnektivität von 8 gelten zusätzlich die vier diagonal zu ihm stehenden Knoten als Nachbarn und die Nachbarschaft wird folglich als ein 3×3 Fenster um den Knoten v verstanden.
Analog zu den Daten der Farbkanäle an den einzelnen Pixeln des Bildes hängt auch an den Knoten über einer Merkmalsfunktion $f\colon \mathcal{V} \to [0, 1]^C$ bzw. einer Merkmalsmatrix $\mathbf{F} \in [0, 1]^{N \times C}$ diese Information mit $f(v) \coloneqq \mathbf{B}_{p(v)_2, p(v)_1}$.

Effiziente Adjazenzbestimmung. Entgegen der Pixelanordnung in einem Bild ist die Anordnung der Knoten in einem Gittergraphen völlig irrelevant und kann willkürlich gewählt werden. Ein Aufbau der Kantenmenge $\mathcal{E}$ bzw. der korrespondierenden, ungewichteten Adjazenzmatrix $\mathbf{A}$ kann jedoch besonders effizient gestaltet werden, wenn die Knoten reihenweise entsprechend ihrer Bildkoordinate angeordnet werden. Dafür wird das Bild $\mathbf{B}$ zunächst an dessen Rändern um eine zusätzliche Spalte bzw. Reihe erweitert, d.h. $\mathbf{B} \in [0, 1]^{(H+2) \times (W+2) \times C}$. Dann besitzt die unge-

wichtete Adjazenzmatrix $\mathbf{A} \in \{0,1\}^{N\times N}$ mit $N \coloneqq (H+2)(W+2)$ in den Reihen $W+3 < i < N-W-3$ an den Spaltenindizes $\{i-W-2, i-1, i+1, i+W+2\}$ bzw.

$$\{i-W-3, i-W-2, i-W-1, i-1, i+1, i+W+1, i+W+2, i+W+3\}$$

genau einen Eintrag gleich Eins (bei einer Konnektivität von 4 bzw. 8). Anschließend müssen die ungültigen Knoten aus $\mathbf{A}$ reihen- und spaltenweise schrittweise über eine Länge von W mit Schrittweite $W+2$ gefiltert werden. Dabei müssen natürlich zunächst die ersten und letzten $W+3$ Knoten aus $\mathbf{A}$ gelöscht werden.

3.2 Superpixel

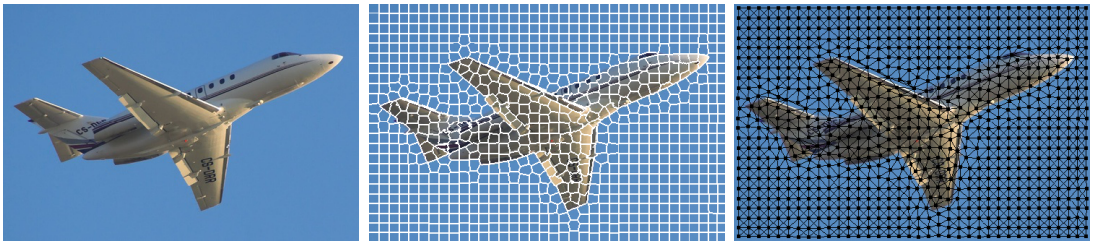
Superpixel erfreuen sich in allen Anwendungen der Bildverarbeitung, insbesondere als Vorverarbeitungsschritt, immer größerer Beliebtheit, da sie die Eingabegröße eines Problems mit oftmals zu vernachlässigem Fehler reduzieren [12]. Superpixel fassen dabei ein Bild in logische, räumliche Gruppen anhand ausgewählter Metriken zusammen, sodass sich Pixel innerhalb einer Gruppe besonders ähnlich sind.
Formal besteht eine *Superpixelrepräsentation* eines Bildes $\mathbf{B} \in \mathbb{R}^{H\times W\times C}$ aus einer Menge von $N \in \mathbb{N}$ *Superpixeln* bzw. *Regionen* $\mathcal{S} \coloneqq \{\mathcal{S}_k\}_{n=1}^{N}$ mit $\mathcal{S}_n \subseteq W \times H$, sodass $\mathcal{S}_i \cap \mathcal{S}_j = \emptyset$ und $\bigcup_{\mathcal{S}_n\in\mathcal{S}} \mathcal{S}_n = W \times H$ [33]. Für eine gültige Superpixelsegmentierung fordern wir desweiteren, dass $\mathcal{S}_n$ zusammenhängend ist [33]. Aus der Menge an Superpixeln $\mathcal{S}$ lässt sich damit eine *Segmentierungsmaske* $\mathbf{S} \in \{1,\dots,N\}^{H\times W}$ über

$$\mathbf{S}_{yx} = n \text{ genau dann, wenn } (x,y) \in \mathcal{S}_n$$

gewinnen, die jedem Pixel in $\mathbf{B}$ eine eindeutige Superpixelzugehörigkeit n zuweist.
Aus einer Superpixelrepräsentation $\mathcal{S}$ bzw. $\mathbf{S}$ eines Bildes $\mathbf{B}$ kann ein Graph im zweidimensionalen euklidischen Raum $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathrm{p})$ wie folgt definiert werden. Seien dafür die einzelnen Superpixel $\mathcal{S} = \{\mathcal{S}_n\}_{n=1}^{N}$ die Knoten $\mathcal{V} = \{v_n\}_{n=1}^{N}$ des Graphen $\mathcal{G}$, d.h. $v_n = \mathcal{S}_n$. Dann ordnet die Positionsfunktion $\mathrm{p}\colon \mathcal{V} \to \mathbb{R}^2$ den einzelnen Superpixeln über

$$\mathrm{p}(v_n) \coloneqq [\bar{x}, \bar{y}]^\top = \frac{1}{|\mathcal{S}_n|} \sum_{(x,y)\in\mathcal{S}_n} [x,y]^\top$$

eine eindeutige Position in der Ebene über deren absoluten Schwerpunkt zu. Die Kantenrelation $\mathcal{E}$ des Graphen $\mathcal{G}$ wird desweiteren über die örtlichen Nachbarregionen der Superpixel basierend auf einem 3×3 Fenster um die Pixel in $\mathbf{S}$ definiert. Falls



(a) Bild (b) Superpixelsegmentierung (c) Graphrepräsentation

Abbildung 3.2: Illustration des Prozesses zur Graphgenerierung anhand einer Superpixelrepräsentation eines Bildes. Ein Bild (a) wird dafür zuerst in eine Menge von Superpixeln segmentiert (b). Die absoluten Zentren der Superpixel bilden die Knotenpunkte des Graphen. Benachbarte Superpixel werden über eine Kante miteinander verbunden (c).

sich beispielsweise die Werte in $\mathbf{S}_{y+1,x}$ und $\mathbf{S}_{y,x}$ unterscheiden, d.h. $\mathbf{S}_{y,x} \neq \mathbf{S}_{y+1,x}$ mit $\mathbf{S}_{y,x} = i$ und $\mathbf{S}_{y+1,x} = j$, dann wird den Knoten $v_i, v_j \in \mathcal{V}$ über $(v_i, v_j) \in \mathcal{E}$ eine Kante in $\mathcal{G}$ zugeordnet. Formal lassen sich dabei erneut die zwei Konnektivitäten 4 und 8 unterscheiden, für die in der Praxis lediglich ein Unterschied in Bereichen erkennbar ist, deren Superpixelregionen rechteckig als Gitter aneinander liegen. Abbildung 3.2 illustriert den beschriebenen Prozess der Graphgenerierung anhand einer Superpixelrepräsentation eines Bildes.
Zusätzlich zu der Positionsfunktion p auf den Superpixelknoten kann der Graph $\mathcal{G}$ um eine *Massefunktion* $\mathrm{m}\colon \mathcal{V} \to \mathbb{N}$ mit $\mathrm{m}(v_n) \coloneqq |S_n|$ zu $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathrm{p}, \mathrm{m})$ erweitert werden, die die Masse bzw. den Flächeninhalt der einzelnen Superpixelregionen beschreibt.

Effiziente Adjazenzbestimmung. Die Bestimmung einer Kantenrelation $\mathcal{E}$ auf einer Superpixelrepräsentation kann aufgrund der pixelweisen Iteration auf der Segmentierungsmaske und dem sukzessiven Vergleich mit den jeweiligen Nachbarspixeln über ein 3×3 Fenster recht teuer werden. Eine effizientere und zugleich elegantere Bestimmung einer Kantenrelation als ungewichtete Adjazenzmatrix kann über eine versetzte Überlagerung der Segmentierungsmasken gewonnen werden. Sei dafür $\mathbf{S} \in \{1, \dots N\}^{H \times W}$ eine Segmentierungsmaske. Dann wird für die vertikale Adjazenzbestimmung die oberste und unterste Reihe aus $\mathbf{S}$ abgeschnitten und in $\mathbf{S}_{\uparrow}, \mathbf{S}_{\downarrow} \in \{1, \dots N\}^{H-1 \times W}$ gespeichert. $\mathbf{S}_{\uparrow}$ und $\mathbf{S}_{\downarrow}$ werden anschließend paarweise auf Ungleichheit getestet und dessen boolsches Ergebnis in einer Binärmaske $(\mathbf{S}_{\uparrow} \neq \mathbf{S}_{\downarrow})$ gespeichert. Liegen dabei wahre, d.h. ungleiche Einträge vor, so ist eine vertikale Adjazenz zwischen zwei unterschiedlichen Regionen vorhanden. Die Einträge $i, j \in \{1, \dots N\}$ in $\mathbf{S}_{\uparrow}$ und $\mathbf{S}_{\downarrow}$ an den Koordinaten der wahren Einträge in der

$$\mathbf{S} = \begin{bmatrix} 1 & 1 & 2 & 3 \\ 1 & 2 & 2 & 3 \\ 1 & 1 & 4 & 4 \end{bmatrix}$$

(a)

$$\mathbf{S}_{\uparrow} = \begin{bmatrix} 1 & 2 & 2 & 3 \\ 1 & 1 & 4 & 4 \end{bmatrix}$$

$$\mathbf{S}_{\downarrow} = \begin{bmatrix} 1 & 1 & 2 & 3 \\ 1 & 2 & 2 & 3 \end{bmatrix}$$

$$(\mathbf{S}_{\uparrow} \neq \mathbf{S}_{\downarrow}) = \begin{bmatrix} \times & \checkmark & \times & \times \\ \times & \checkmark & \checkmark & \checkmark \end{bmatrix}$$

(b)

$$(2,1) \in \mathcal{E}, \quad (1,2) \in \mathcal{E}$$
$$(4,2) \in \mathcal{E}, \quad (4,3) \in \mathcal{E}$$

$$\left(\mathbf{A} \vee \mathbf{A}^{\top}\right) = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix}$$

(c)

Abbildung 3.3: Adjazenzbestimmung einer Segmentierungsmaske $\mathbf{S} \in \{1,2,3,4\}^{3\times 4}$ mit eingezeichneten vertikalen Adjazenzen (a). Daraus lassen sich $\mathbf{S}_{\uparrow}$, $\mathbf{S}_{\downarrow}$ sowie $\mathbf{S}_{\uparrow} \neq \mathbf{S}_{\downarrow}$ bestimmen (b). Die Einträge in $\mathbf{S}_{\uparrow}$ und $\mathbf{S}_{\downarrow}$ an den Indizes der wahren Werte in $\mathbf{S}_{\uparrow} \neq \mathbf{S}_{\downarrow}$ bilden jeweils eine Kante im Graphen (c).

Binärmaske bilden folglich die Knotenindizes einer Kante $(v_i, v_j) \in \mathcal{E}$ benachbarter Regionen und werden in eine Adjazenzmatrix mit $\mathbf{A}_{ij} = 1$ übertragen. Analog wird für die horizontale Adjazenzbestimmung vorgegangen. Die Veroderung $\mathbf{A} \vee \mathbf{A}^{\top}$ liefert dann die resultierende ungerichtete, ungewichtete Adjazenzmatrix mit einer Konnektivität von 4 (vgl. [31]). Abbildung 3.3 illustriert das Verfahren anhand eines einfachen Beispiels.

Globale und lokale Normierung. Eine Menge generierter Graphen aus den Superpixelrepräsentationen einer Bildermenge können sich zu Teilen extrem unterscheiden, beispielsweise durch variierende Auflösungen der Bilder. So besitzt ein Graph möglicherweise nur sehr „kurze“ Kanten, wohingegen ein anderer ausschließlich im Vergleich dazu aus „längeren“ Kanten besteht. Das kann bei der Transformation solcher Graphen in die Adjazenzmatrixrepräsentation $\mathbf{A}_{\text{dist}} \in [0,1]^{N\times N}$ aufgrund der Wahl des Parameters $\xi \in \mathbb{R}$ zu Problemen führen. Eine statische Wahl des Parameters ξ führt dann gegebenenfalls zu Einträgen in $\mathbf{A}_{\text{dist}}$, die stets sehr nahe bei Eins bzw. Null liegen. Eine Normierung der Kantenlängen erscheint daher für die Transformation nach $\mathbf{A}_{\text{dist}}$ sinnvoll. Eine *Normierung* eines Graphen $\mathcal{G}$ sieht dafür bei der Transformation nach $\mathbf{A}_{\text{dist}}$ eine Skalierung von $\|\mathrm{p}(v_i) - \mathrm{p}(v_j)\|_2^2 \in \mathbb{R}$ in das Intervall $[0,1]$ vor. Eine *globale Normierung* von $\mathcal{G}$ skaliert dabei $\|\mathrm{p}(v_i) - \mathrm{p}(v_j)\|_2^2$ über die maximale Kantenlänge $\|\cdot\|_2^2$ aller Kanten des Graphen, wohingegen die *lokale Normierung* lediglich eine Skalierung über die maximale Kantenlänge der ausgehenden Kanten von $v_i \in \mathcal{V}$ vorsieht. Durch die lokale Normierung ist die Adjazenzmatrix $\mathbf{A}_{\text{dist}}$ insbesondere gerichtet. Ein üblicher Trick, um dem entgegenzuwirken, ist die Aufsummierung von $\mathbf{A}_{\text{dist}}$ mit ihrer transponierten Matrix $\mathbf{A}_{\text{dist}}^{\top}$ und der anschließenden Halbierung ihrer Werte, d.h. $\frac{1}{2}\left(\mathbf{A}_{\text{dist}} + \mathbf{A}_{\text{dist}}^{\top}\right)$ [26].

Es ist anzumerken, dass der normierte Graph, repräsentiert als $\mathbf{A}_{\text{dist}}$, neben der bereits vorhandenen Translationsinvarianz insbesondere skalierungsinvariant ist. Im weiteren Verlauf dieser Arbeit wird der Prozess der Normierung eines Graphen implizit angenommen.

3.2.1 Verfahren

In der Literatur finden sich zahlreiche Verfahren zur Bestimmung einer Superpixelrepräsentation aus einem Bild mit jeweils unterschiedlichen Stärken und Schwächen [2, 33]. Zwei beliebte Verfahren, die aufgrund ihrer im Allgemeinen guten Resultate bei geringer Berechnungskomplexität immer wieder zu finden sind, sind der SLIC- sowie der Quickshift-Algorithmus [2, 11, 12, 16, 33]. Beide Verfahren werden im Folgenden näher vorgestellt.

SLIC. Der Superpixelalgorithmus *Simple Linear Iterative Clustering (SLIC)* ist ein recht einfach gehaltener lokaler *K-Means*-Clustering-Ansatz, welcher sich dennoch durch seine Geschwindigkeit, seine Speichereffizienz und insbesondere durch seine erfolgreiche Segmentierung hinsichtlich der Farbabgrenzen seiner Superpixel im Vergleich zu anderen „State-of-the-Art"-Algorithmen auszeichnet [2, 12].
Ein Bild $\mathbf{B} \in \mathbb{R}^{H \times C \times 3}$ im *Lab-Farbraum* wird dazu initial in $K \in \mathbb{N}$ viele Clusterzentren $\mathbf{c}_k \coloneqq [l_k, a_k, b_k, x_k, y_k]^\top$ mit jeweils gleichmäßigem Abstand $S \coloneqq \sqrt{WH/K}$ zerlegt [2]. Daraufhin werden in einem iterativen Prozess die Zugehörigkeiten der Pixel zu ihren jeweiligen Clustern über eine Distanzmetrik D bestimmt und die Clusterzentren entsprechend ihrer neuen Gruppierungen angepasst. Entgegen der konventionellen k-Means-Formulierung werden dabei aber nicht die Distanzen jedes Pixels zu jedem Clusterzentrum berechnet, sondern lediglich für die Pixel, die sich in der Region der Größe $2S \times 2S$ um $\mathbf{c}_k$ befinden [2]. Dies führt folglich zu einer drastischen Reduzierung der Anzahl an Distanzberechnungen und zu einem effizienteren Algorithmus. Wurde zu jedem Pixel dessen ähnlichstes Clusterzentrum bestimmt, werden die Zentren über die Durchschnittsbildung all seiner zugehörigen Pixel neu justiert. Nach einer festgelegten Anzahl an Iterationsschritten (üblicherweise 10) bricht der Algorithmus schließlich ab [2]. Das gesamte Verfahren ist in Algorithmus 3.1 zusammengefasst.
Die Distanzmetrik D basiert auf der euklidischen Norm $\|\cdot\|_2$ im fünfdimensionalen Raum $[l, a, b, x, y]^\top$ auf den Farben und den Positionen der Pixel bzw. der Clusterzentren. Dabei ergeben sich jedoch Inkonsistenten für unterschiedliche Superpixelgrößen. Bei großen Superpixeln führt dies zu einer stärkeren Gewichtung der

Eingabe: Bild $\mathbf{B} \in \mathbb{R}^{H \times W \times 3}$ im Lab-Farbraum, Anzahl der Cluster K
Ausgabe: Segmentierungsmaske $\mathbf{S} \in \mathbb{N}^{H \times W}$
  Initialisiere K Clusterzentren $\{\mathbf{c}_k\}_{k=1}^{K}$ mit gleichmäßigem Abstand S.
  $\mathbf{S}_{yx} \leftarrow -1$ für jedes Pixel $\mathbf{B}_{yx}$ an (x, y).
  $\mathbf{D}_{yx} \leftarrow \infty$ für jedes Pixel $\mathbf{B}_{yx}$ an (x, y).
  repeat
    for $\mathbf{c}_k \in \{\mathbf{c}_k\}_{k=1}^{K}$ **do**
      for Pixel $\mathbf{B}_{yx}$ an (x, y) in $2S \times 2S$ Region um $\mathbf{c}_k$ **do**
        Berechne Distanz D zwischen $\mathbf{c}_k$ und $\mathbf{B}_{yx}$.
        if $D < \mathbf{D}_{yx}$ **then**
          $\mathbf{D}_{yx} \leftarrow D$
          $\mathbf{S}_{yx} \leftarrow k$
        end if
      end for
    end for
    Justiere Clusterzentren $\{\mathbf{c}_k\}_{k=1}^{K}$.
  until Endbedingung

Algorithmus 3.1: SLIC-Algorithmus, der eine Segmentierungsmaske $\mathbf{S} \in \mathbb{N}^{H \times W}$ in den Ausmaßen des Eingabebildes $\mathbf{B} \in \mathbb{R}^{H \times W \times 3}$ über ein lokales K-Means-Clustering bei K gleichmäßig verteilten initialen Clusterzentren generiert.

räumlichen Positionen der Superpixel, wohingegen bei kleinen Superpixeln genau das Gegenteil der Fall ist [2]. Es ist daher notwendig, die Distanzen der Positionen D_S und Farben D_F getrennt voneinander zu betrachten und mittels dem Abstand S der initialen Clusterzentren und einer Normalisierungskonstante $F \in \mathbb{R}$ bezüglich der Gewichtung der Farben zu normieren. Damit ergibt sich D als [2]

$$
\begin{aligned}
D &\coloneqq \sqrt{\left(\frac{D_S}{S}\right)^2 + \left(\frac{D_F}{F}\right)^2} \\
D_S &\coloneqq \left\| [x, y]^\top - [x_k, y_k]^\top \right\|_2 \\
D_F &\coloneqq \left\| [l, a, b]^\top - [l_k, a_k, b_k]^\top \right\|_2.
\end{aligned}
$$

Die Normalisierungskonstante F kann damit auch als Gewichtung zwischen der Form und den Farbabgrenzungen der Superpixel verstanden werden. Falls F sehr klein gewählt wird, respektieren die Superpixel Farbabgrenzungen besser, aber besitzen im Allgemeinen auch eine eher unreguläre Form [2].

Damit ist SLIC ein $\mathcal{O}(WH)$ effizienter Algorithmus mit nur zwei frei wählbaren Parametern K und F [2]. Viele Anwendungen im Bereich der Bildverarbeitung machen sich daher SLIC zu Nutze (vgl. [12, 16, 33]). Abbildung 3.4 (a) zeigt ein

(a) SLIC (b) Quickshift

Abbildung 3.4: Ein Bus segmentiert über SLIC mit jeweils 400, 800 und 1600 Superpixeln (a) sowie über Quickshift mit 600 Superpixeln (b). Dabei werden die unterschiedlichen Verfahren zur Generierung von Superpixeln deutlich. Wohingegen SLIC möglichst quadratische, gleichgroße Superpixel erzeugt, generiert Quickshift sowohl sehr große wie auch sehr kleine Superpixel in allen möglichen Formvariationen.

Beispielresultat des SLIC-Algorithmus bei unterschiedlich gewählten Anzahlen an Clusterzentren.

Quickshift. Ein weiteres Verfahren zur Bildsegmentierung ist der gradientenbasierte Algorithmus *Quickshift*, der unter anderem bereits zur Objektlokalisierung auf Basis von Superpixeln benutzt wurde [11, 34].
Quickshift startet dabei mit der Berechnung der *Parzen-Dichteschätzung* p eines Bildes $\mathbf{B} \in \mathbb{R}^{H \times W \times 3}$ für jedes Pixel $\mathbf{B}_{yx} \in \mathbb{R}^3$ im fünfdimensionalen Raum über die Gaußfunktion, d.h.

$$p(x, y, \mathbf{B}) = \sum_{\substack{1 \leq i \leq W \\ 1 \leq j \leq H}} \frac{1}{(2\pi\xi)^5} \exp\left(-\frac{1}{2\xi^2} \left\| \begin{pmatrix} x - x_i \\ y - y_i \\ \alpha(\mathbf{B}_{yx} - \mathbf{B}_{y_i x_i}) \end{pmatrix} \right\|_2^2 \right),$$

mit der Standardabweichung $\xi \in \mathbb{R}$ und der Gewichtung des Farbeinflusses über $\alpha \in \mathbb{R}$ [34]. Je größer ξ gewählt wird, umso größer wird der Anteil weitentfernter Pixel zu der Dichte des jeweils betrachteten Pixels. In der Praxis kann, analog wie bei SLIC, eine Fenstergröße $S \times S$ zur Einschränkung der Betrachtung der Pixel um einen Pixel definiert werden [33]. Obwohl Quickshift standardmäßig S auf ∞ setzt, lohnt es sich aus Effizienzgründen eine Größe in Abhängigkeit zu ξ zu wählen [34].
Basierend auf den Dichten des Bildes generiert Quickshift einen Baum, der jedem

Pixel $\mathbf{B}_{y_i x_i}$ einen Nachbarspixel $\mathbf{B}_{y_j x_j}$ zuordnet, welcher einen höheren Dichtewert besitzt, d.h.

$$p(x_j, y_j, \mathbf{B}) > p(x_i, y_i, \mathbf{B}).$$

Falls $\mathbf{B}_{y_i x_i}$ kein solches Pixel zugeordnet werden kann, wird es mit keinem Pixel verbunden. Die verbundene Menge an Knoten repräsentiert dann schließlich einen Superpixel des Bildes.
Quickshift segmentiert damit ein Bild auf Basis von drei Parametern — ξ für die Standardabweichung der Gaußfunktion, α zur Gewichtung des Farbterms sowie S zur Einschränkung der Berechnung über ein Fenster der Größe $S \times S$.
Die Superpixel, die Quickshift produziert, besitzen im Gegensatz zu den Superpixeln in SLIC keine Einschränkung in ihrer Anzahl, Form oder Größe. Ein segmentiertes Bild mit feinen Strukturen besitzt dann wohlmöglich weitaus mehr Superpixel, als ein Bild mit großen gleichfarbigen Flächen, welches nur sehr wenige, großflächige Superpixel generiert. Ebenso gibt es im Gegensatz zu SLIC keinen Parameter, der die Form der Superpixel steuern kann, sodass Quickshift Superpixel aller erdenklichen Formen produzieren kann. Abbildung 3.4 (b) illustriert ein Resultat einer Quickshift-Segmentierung und insbesondere den Vergleich zu dem entsprechenden SLIC-Äquivalent.

Weitere Verfahren. In den letzten Jahren wurden desweiteren zahlreiche eigenvektorbasierte Verfahren zur Bildsegmentierung entwickelt [10, 33]. Obwohl diese vielversprechende Resultate aufweisen, sind sie jedoch entsprechend langsam um von praktischem Nutzen für die meisten Anwendungen zu sein [10]. Andere Algorithmen wiederum zeigen sich als besonders berechnungseffizient, liefern aber dementsprechend unzufriedenstellende Ergebnisse [10]. Namenshaft zu erwähnen sei noch die *effiziente graphbasierte Bildsegmentierung* von Felzenszwalb und Huttenlocher, die in der Regel unter dem Namen *Felzenszwalb-Segmentierung* bekannt ist [10]. Der Algorithmus von Felzenszwalb und Huttenlocher zeichnet sich dabei durch seine geringe Berechnungskomplexität aus und versucht gleichzeitig, globale Eigenschaften des Bildes zu erhalten. Dafür wird das Bild als regulärer Gittergraph initialisiert, bei dem die Kantengewichte eine Aussage über den Unterschied in Helligkeit und Farbe der benachbarten Pixel treffen. Der Algorithmus versucht daraufhin, Knoten insofern zu verschmelzen, dass der Unterschied zwischen den Kantengewichtungen innerhalb einer Region möglichst gering bleibt und zwischen benachbarten Regionen möglichst groß wird (vgl. [10]). Die Felzenszwalb-Segmentierung zeigte sich aber in Tests auf einer Reihe von Bildern weniger geeignet, da dessen Parameter für je-

des Bild eine spezielle Anpassung benötigen und folglich eine statische Wahl dieser Parameter zu unbrauchbaren Ergebnissen führt.

3.2.2 Merkmalsextraktion

Die Darstellung eines Bildes über einen Graphen $\mathcal{G}$, der aus einer Superpixelrepräsentation $\mathcal{S}$ gewonnen wurde, besitzt in der Regel weitaus weniger Knoten im Gegensatz zu der reinen Darstellung des Bildes über eine Gitterrepräsentation. Die Superpixel bzw. die Regionen der Segmentierungsmaske können dabei jedoch die willkürlichsten Formen annehmen und besitzen lediglich die Einschränkung, dass diese stets zusammenhängend sind. Die Form eines Superpixels muss demnach bestmöglichst eingefangen bzw. beschrieben werden können — ein Prozess, der in der Bildverarbeitung als *Merkmalsextraktion* bekannt ist [32]. Ein geeignetes Mittel zur Beschreibung einzelner Objekte in einem segmentierten Bild sind die *Momente*, welche in nicht-zentrierte, translationsinvariante, skalierungsinvariante und rotationsinvariante Momente unterschieden werden [32].

Nicht-zentrierte Momente. Zu der binären Segmentierungsmaske $\mathbf{S} \in \{0,1\}^{H \times W}$ sind die *nicht-zentrierten Momente* vom Grad $(i+j)$, $i, j \in \mathbb{N}$, definiert als [32]

$$\mathbf{M}_{ij} \coloneqq \sum_{x}^{W} \sum_{y}^{H} x^i y^j \mathbf{S}_{yx}.$$

Obwohl der Grad eines Moments beliebig hoch gewählt werden kann, so reichen in der Praxis meist wenige Momente niedrigen Grades aus (≤ 3), um eine Region hinreichend genau zu charakterisieren [32]. Bildeigenschaften, die durch nicht-zentrierte Momente beschrieben werden können, sind unter anderem dessen Fläche über $\mathbf{M}_{00}$ sowie dessen absoluter Schwerpunkt $\{\bar{x}, \bar{y}\} = \{\mathbf{M}_{10}/\mathbf{M}_{00}, \mathbf{M}_{01}/\mathbf{M}_{00}\}$ [32].

Translationsinvariante (zentrale) Momente. Nicht-zentrierte Momente sind aufgrund ihrer Berücksichtigung der Position einer Region im Bild meist unerwünscht, sie helfen aber für die weitere Definition von translationsinvarianten Momenten. Mit Hilfe der absoluten Schwerpunktskoordinaten $\{\bar{x}, \bar{y}\}$ können die *translationsinvarianten Momente* über

$$\mu_{ij} \coloneqq \sum_{x}^{W} \sum_{y}^{H} (x - \bar{x})^i (y - \bar{y})^j \mathbf{S}_{yx}.$$

definiert werden [32]. Sie lassen sich weiterhin direkt aus $\mathbf{M}_{ij}$ ermitteln. So gilt zum Beispiel, dass $\mu_{00} = \mathbf{M}_{00}$ oder $\mu_{11} = \mathbf{M}_{11} - \bar{x}\mathbf{M}_{01} = \mathbf{M}_{11} - \bar{y}\mathbf{M}_{10}$ (vgl. [32]).

Skalierungsinvariante Momente. Für $i + j \geq 2$ können desweiteren die *skalierungsinvarianten Momente* η_{ij} konstruiert werden, die invariant bezüglich Skalierung und Translation sind. Dafür wird das entsprechende translationsinvariante Moment μ_{ij} durch die entsprechende Fläche $\mathbf{M}_{00}$ bzw. μ_{00} des Segments geteilt, d.h. [32]

$$\eta_{ij} = \frac{\mu_{ij}}{\mu_{00}^{(1+(i+j)/2)}}.$$

(Rotationsinvariante) Hu-Momente. Hu verwendet eine nichtlineare Kombination der skalierungsinvarianten Momente bis zum Grad 3, um aus ihnen zusätzlich eine Rotationsinvarianz zu gewinnen [18, 32]. Daraus ergeben sich die sieben *rotationsinvariante Momente* bzw. die *Hu-Momente* [32].

$$\begin{aligned}
\mathbf{h}_1 &= \eta_{20} + \eta_{02} \\
\mathbf{h}_2 &= (\eta_{20} - \eta_{02})^2 + (2\eta_{11})^2 \\
\mathbf{h}_3 &= (\eta_{30} - 3\eta_{12})^2 + (3\eta_{21} - \eta_{03})^2 \\
\mathbf{h}_4 &= (\eta_{30} + \eta_{12})^2 + (\eta_{21} + \eta_{03})^2 \\
\mathbf{h}_5 &= (\eta_{30} - 3\eta_{12})(\eta_{30} + \eta_{12})\Big((\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2\Big) \\
&\quad + (3\eta_{21} - \eta_{03})(\eta_{21} + \eta_{03})\Big(3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2\Big) \\
\mathbf{h}_6 &= (\eta_{20} - \eta_{02})\Big((\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2\Big) + 4\eta_{11}(\eta_{30} + \eta_{12})(\eta_{21} + \eta_{03}) \\
\mathbf{h}_7 &= (3\eta_{21} - \eta_{03})(\eta_{30} + \eta_{12})\Big((\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2\Big) \\
&\quad + (\eta_{03} - 3\eta_{12})(\eta_{21} + \eta_{03})\Big(3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2\Big)
\end{aligned}$$

Weitere Merkmale. Aus den translationsinvarianten Momenten lassen sich weitere Merkmale gewinnen, denen insbesondere eine anschauliche Interpretation zu Grunde liegt. Sei dafür die *Kovarianzmatrix* der binären Segmentierungsmaske definiert über

$$\begin{bmatrix} \mu'_{20} & \mu'_{11} \\ \mu'_{11} & \mu'_{02} \end{bmatrix},$$

wobei $\mu'_{20} \coloneqq \mu_{20}/\mu_{00}$, $\mu'_{02} \coloneqq \mu_{02}/\mu_{00}$ und $\mu'_{11} \coloneqq \mu_{11}/\mu_{00}$ [32]. Die beiden Eigenvektoren dieser Matrix

$$\lambda_i = \frac{\mu'_{20} + \mu'_{02}}{2} \mp \frac{\sqrt{4\mu'_{11} + (\mu'_{20} - \mu'_{02})^2}}{2}$$

entsprechen der Länge der großen bzw. kleinen Halbachse einer Ellipse, die die Region minimal umschließt [32]. Damit kann die *Ausrichtung* oder *Orientierung* der Region aus dem Winkel des Eigenvektors des größten Eigenwerts mit Hilfe des Arkustangens über $\text{ori} \coloneqq \text{atan}(2\mu'_{11}/(\mu'_{20} - \mu'_{02}))/2$ berechnet werden. Ähnlich dazu lässt sich die *Exzentrizität* $\text{ecc} \coloneqq \sqrt{1 - \lambda_1/\lambda_2}$ als das Verhältnis der beiden Hauptachen zueinander definieren [29].
Neben den Merkmalen, die sich aus den Momenten ergeben, können noch zahlreiche weitere Merkmale aus den Formen einer Region gewonnen werden (vgl. [29]). Beispiele dafür sind unter anderem die Merkmale des *minimalen Hüllkörpers* (engl. *Bounding-Box*) einer Region, d.h. dem kleinsten waagerechten Rechteck, welches die Region umschließt. Die Breite, Höhe und Fläche dieses Hüllkörpers können aus einer binären Segmentierungsmaske $\mathbf{S} \in \{0,1\}^{H \times W}$ über

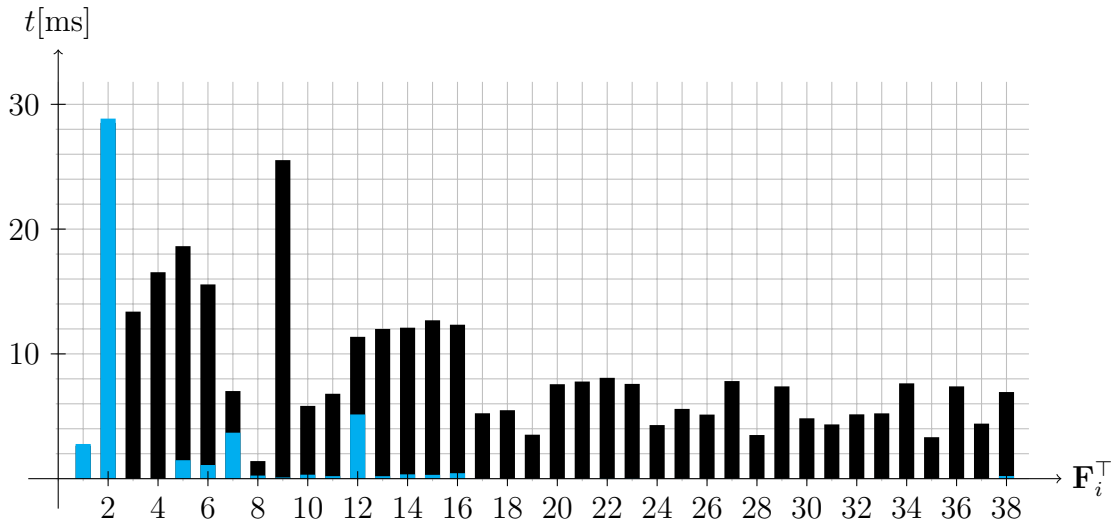
$$\begin{aligned} \text{box}_x &\coloneqq \max(\{x \mid \mathbf{S}_{yx} = 1\}) - \min(\{x \mid \mathbf{S}_{yx} = 1\}) \\ \text{box}_y &\coloneqq \max(\{y \mid \mathbf{S}_{yx} = 1\}) - \min(\{y \mid \mathbf{S}_{yx} = 1\}) \\ \text{box} &\coloneqq \text{box}_x \text{box}_y \end{aligned}$$

gewonnen werden. Das *Ausmaß* einer Region (engl. *Extent*) ist weiterhin definiert als $\text{ext} \coloneqq \mathbf{M}_{00}/\text{box}$ [29]. Der *gleichwertige Durchmesser* (engl. *Equivalent-Diameter*) zu der Fläche einer Region lässt sich über $\text{dia} \coloneqq \sqrt{4\mathbf{M}_{00}/\pi}$ beschreiben [29]. Abschließend lassen sich die absoluten Schwerpunktskoordianten $\{\bar{x}, \bar{y}\}$ in relative, d.h. translationsinvariante, Schwerpunktskoordinaten $\{\hat{x}, \hat{y}\}$ mittels

$$\hat{x} \coloneqq \bar{x} - \min(\{x \mid \mathbf{S}_{yx} = 1\}) \qquad \hat{y} \coloneqq \bar{y} - \min(\{y \mid \mathbf{S}_{yx} = 1\})$$

überführen. Insgesamt ergeben sich daraus 38 Merkmale, die die Form eines einzelnen Superpixels beschreiben (vgl. Abbildung 3.5). Zusätzlich zu diesen beschreiben drei weitere Merkmale die Durchschnittsfarbe der drei Farbkanäle eines Superpixels.

Caching. Viele der 38 ermittelten Merkmale werden über bereits definierte Merkmale beschrieben. So bauen insbesondere die Momente sukzessive über die zusätzlichen Stufen der Invarianz aufeinander auf. Weiterhin erfordern Merkmale wie die Orientierung oder die Exzentrizität gleichermaßen die Berechnung der translationsinvarianten Momente sowie dessen Kovarianzmatrix. Auch bei Merkmalen, die nicht über die Momente beschrieben sind, wie zum Beispiel die Berechnung des minimalen Hüllkörpers und der relativen Schwerpunktskoordinaten, lassen sich Gemeinsamkeiten in der Berechnung wiederfinden, zum Beispiel über die Eckpunktkoordinaten des



1. $\mathbf{M}_{00}$	2. box	3. box_x	4. box_y	5. $\hat{x}$	6. $\hat{y}$	7. ecc	8. dia
9. ext	10. $\mathbf{h}_1$	11. $\mathbf{h}_2$	12. $\mathbf{h}_3$	13. $\mathbf{h}_4$	14. $\mathbf{h}_5$	15. $\mathbf{h}_6$	16. $\mathbf{h}_7$
17. η'_{02}	18. η'_{11}	19. η'_{20}	20. λ_1	21. λ_2	22. axis_1	23. axis_2	24. μ_{02}
25. μ_{03}	26. μ_{11}	27. μ_{12}	28. μ_{20}	29. μ_{21}	30. μ_{30}	31. η_{02}	32. η_{03}
33. η_{11}	34. η_{12}	35. η_{20}	36. η_{21}	37. η_{30}	38. ori		

Abbildung 3.5: Laufzeitverteilung einer Extraktion aller 38 Form-Merkmale einer SLIC-Segmentierung mit 800 Regionen. Die schwarzen Balken kennzeichen die einzelnen Berechnungskomplexitäten der Merkmale. Die blauen Balken hingegen demonstrieren den Laufzeitgewinn mittels Caching bei der Berechnung aller Merkmale. Es zeigt sich, dass bereits nach der Berechnung des zweiten Merkmals ein deutlicher Geschwindigkeitsgewinn zu vernehmen ist.

Hüllkörpers. Obwohl eine Berechnung aller Merkmale aufgrund der stetigen Wiederverwertung der Daten redundant erscheint und die Menge der zu benutzenden Merkmale für ein spezifisches Problem auf eine Untermenge reduziert werden sollte (vgl. Kapitel 6.2), zeigt es sich dennoch als lohnenswert, alle bereits errechneten Daten für die weiteren Berechnungen zu cachen. Abbildung 3.5 illustriert dabei den enormen Laufzeitgewinn gegenüber einer Berechnung ohne Verwendung eines Caches. Es ist jedoch anzumerken, dass der Laufzeitgewinn mit den Kosten eines erhöhten Speicheraufwands verbunden ist.

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 euklidischen 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. [28]). 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. [4]).

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 [17]. 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. $\mathbf{M} = \mathbf{M}^\top$ [17]. 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 [17]. 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. $\mathbf{\Lambda}_{ii} = \lambda_i$ [6]. Dann gilt $\mathbf{MU} = \mathbf{U\Lambda}$ und insbesondere ist $\mathbf{M}$ diagonalisierbar über [17]

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

Weiterhin gilt für die k-te Potenz von $\mathbf{M}$, $k \in \mathbb{N}$, [20]

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

aufgrund 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 $\mathbf{x} \in \mathbb{R}^N$ [17]. Eigenwerte symmetrischer positiv semidefiniter Matrizen $\lambda_i \in \mathbb{R}_+$ sind positiv reell und es lässt sich folglich auf diesen eine Ordnung definieren mit $0 \leq \lambda_1 \leq \dots \leq \lambda_N \coloneqq \lambda_{\max}$ [17].

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 [4]. 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 [4]. 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 $\mathbf{L} \coloneqq \mathbf{D} - \mathbf{A}$ [4]. 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. $\mathbf{D}_{ii} = 0$ [4]. Daraus ergibt sich die elementweise Definition

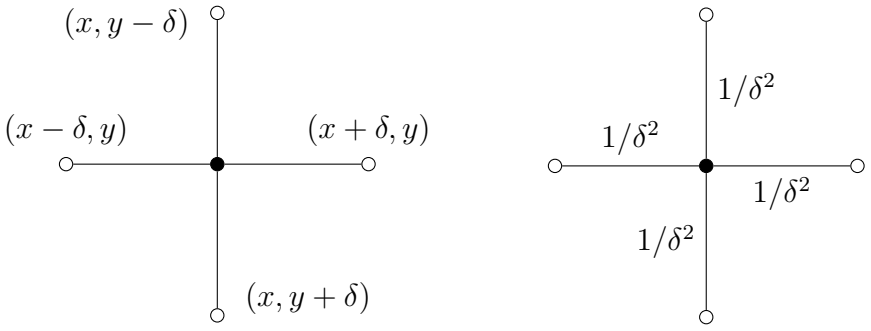
$$\tilde{\mathbf{L}}_{ij} \coloneqq \begin{cases} 1, & \text{wenn } i = j, \\ -\frac{w(v_i, v_j)}{\sqrt{d(v_i)d(v_j)}}, & \text{wenn } v_j \in \mathcal{N}(v_i), \\ 0, & \text{sonst.} \end{cases}$$

Für zusammenhängende Graphen kann $\tilde{\mathbf{L}}$ vereinfacht werden zu [4]

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

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 [26]. $\mathbf{L}$ 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 [13]. 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 [13]. 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 $\{(x, y), (x+\delta, y), (x-\delta, y), (x, y+\delta), (x, y-\delta)\}$ [13] (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 [13]. 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 [4]. 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 $\mathbf{L}\mathbf{u}_1 = \mathbf{0}$ [28]. $\tilde{\mathbf{L}}$ hingegen ist nicht zwingend schwach diagonaldominant. Es lässt sich jedoch zeigen, dass auch für $\tilde{\mathbf{L}}$ gilt, dass $\lambda_1 = 0$ [4].
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 [4]. 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 [4].
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 $s(v_i, v_j) > k$ [13]. 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 [28]. 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. [28]). 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 [28]

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

Die komplexen Exponentiale $e^{2\pi i\omega t}$ beschreiben dabei die Eigenfunktionen des eindimensionalen Laplace-Beltrami Operators [28]:

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

$\hat{f}$ kann damit als die Ausdehnung von f in Bezug auf die Eigenfunktionen des Laplace-Beltrami Operators ∇^2 verstanden werden [13].

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 [28]:

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

Die inverse Graph-Fourier-Transformation ergibt sich dann als [28]

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

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. [28]).

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 [28]. $\hat{\mathbf{g}} \in \mathbb{R}^N$ ist damit ein *nicht-parametrischer* Filter, d.h. ein Filter, dessen Werte für alle Frequenzen $\{\lambda_n\}_{n=1}^N$ frei wählbar sind [6]. 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 [6, 28]

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

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 $\mathcal{O}(N^2)$ [6]. Zudem muss $\mathbf{U}$ zuerst bestimmt werden — ein kostspieliger Aufwand für Graphen mit möglicherweise weit mehr als hundert Knoten [20]. 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 [6]. 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 [6, 13]. Die Filtergröße sinkt somit auf einen konstanten Faktor K mit Lernaufwand $\mathcal{O}(K)$, dem gleichen Aufwand klassischer zweidimensionaler CNNs [6]. $\mathbf{f}_{\text{in}} \star \hat{\mathbf{g}}$ ergibt dann nach (5.1), (5.9) und (5.10) approximiert durch [6]

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

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 [13].

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 [28].

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. [13]). 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 $T_1(x) = x$ [13]. Ein Tschebyschow-Polynom T_k ist ein Polynom k-ten Grades und liegt im Intervall $[-1, 1]$ für $x \in [-1, 1]$ [13]. 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 [6]. 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 [6]. $\mathbf{U} T_k\Big(\tilde{\mathbf{\Lambda}}\Big) \mathbf{U}^\top$ 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 [6]. 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 [6]. 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. [13]). $\mathbf{f}_{\text{out}}$ 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 $\mathcal{O}(K|\mathcal{E}|)$ [6].

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 [20].

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 [20]. 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 bewährt, 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. [15, 20, 30]). 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 [20]. 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 [20]. 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 [20]

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

Die skalierten Eigenwerte von $\tilde{\mathbf{\Lambda}}$ liegen aufgrund der Addition mit $\mathbf{I}$ nun im Intervall $[0, 2]$ (vgl. [20]). Demnach können wiederholte Anwendungen des Faltungsoperators zu „numerischen Instabilitäten und folglich zu explodierenden oder verschwindenen Gradienten" führen [20]. 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 $v_i \in \mathcal{N}(v_i)$ [35]. 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 [20].

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 [8]. Basierend auf einer initialen kontinuierlichen 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

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

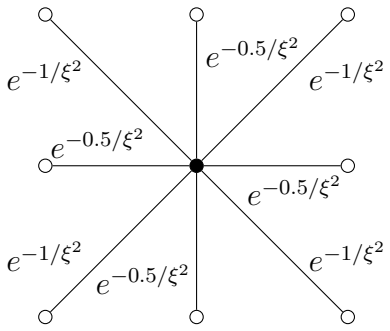
wobei $1/\sqrt{d_i d_j} \in \mathbb{R}$ die Normalisierungskonstante für die Kante $(v_i, v_j) \in \mathcal{E}$ entsprechend der Normalisierung $\tilde{\mathbf{D}}^{-1/2}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-1/2}$ aus (5.16) und $\mathbf{W}^{(t)} \in \mathbb{R}^{N \times N}$ die Filtermatrix der t-en Faltungsschicht beschreibt [20]. Folglich kann die Faltung des GCNs als „differenzierbare und parametrisierte Generalisierung des eindimensionalen Weisfeiler-Lehman-Algorithmus auf Graphen“ verstanden werden [20].

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. [6, 20]). 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 $K = 3$ [20].
Die spektrale Faltung auf Graphen entspricht einer Generalisierung der Faltung klassischer CNNs auf zweidimensionalen Bildern [19]. 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 [6].
Im Kontext dieser Arbeit, dem Lernen auf Graphen im zweidimensionalen euklidischen 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

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

(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 mit $\exp(-0.5/\xi^2) \in \mathbb{R}$ bzw. mit $\exp(-1/\xi^2) \in \mathbb{R}$ bei den Diagonalen.

$\tilde{\mathbf{D}}^{-1/2}$) effektiv zu einer Filtermatrix der Form

$$\begin{bmatrix} ce^{-1/\xi^2} & ce^{-0.5/\xi^2} & ce^{-1/\xi^2} \\ ce^{-0.5/\xi^2} & c & ce^{-0.5/\xi^2} \\ ce^{-1/\xi^2} & ce^{-0.5/\xi^2} & ce^{-1/\xi^2} \end{bmatrix} = c \begin{bmatrix} e^{-1/\xi^2} & e^{-0.5/\xi^2} & e^{-1/\xi^2} \\ e^{-0.5/\xi^2} & 1 & e^{-0.5/\xi^2} \\ e^{-1/\xi^2} & e^{-0.5/\xi^2} & e^{-1/\xi^2} \end{bmatrix}$$

mit einem einzigen trainierbaren Parameter $c \in \mathbb{R}$ bei horizontalen bzw. vertikalen Kantengewichten nach (3.1) mit $\exp(-0.5/\xi^2) \in \mathbb{R}$ bzw. mit $\exp(-1/\xi^2) \in \mathbb{R}$ 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. [19]). 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 euklidischen Raum äquivalent zu der klassischen Formulierung auf regulären Gittern verhält.

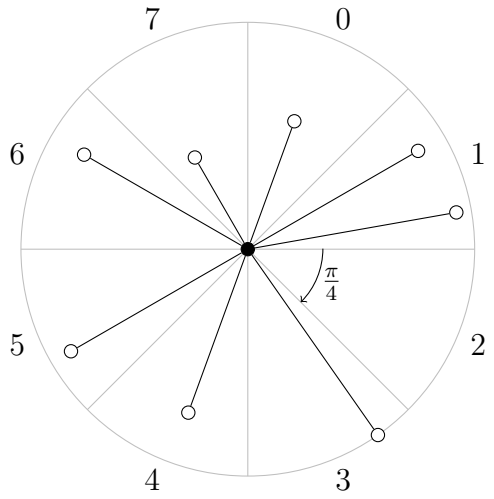


Abbildung 5.3: Partitionierung eines Graphknotens im Uhrzeigersinn in $P = 8$ Bereiche mit gleichmäßigen Innenwinkeln der Größe $\pi/4$.

5.4.1 Partitionierung

Sei $\mathcal{G}$ ein Graph im zweidimensionalen euklidischen Raum, 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 euklidischen 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 $\mathbf{d}_0, \ldots, \mathbf{d}_M \in \mathbb{R}^d$ [5]. 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 $(k-1 \to k)$

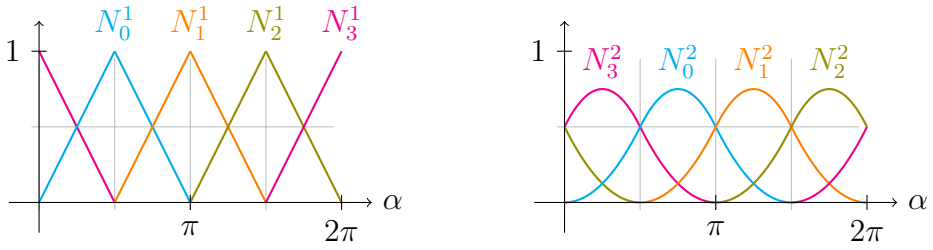
$$N_m^k(t) \coloneqq \frac{t - t_m}{t_{m+k} - t_m} N_m^{k-1}(t) + \frac{t_{m+k+1} - t}{t_{m+k+1} - t_{m+1}} N_{m+1}^{k-1}(t)$$

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 $t \in (t_0, t_{M+1}]$ [5].

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.

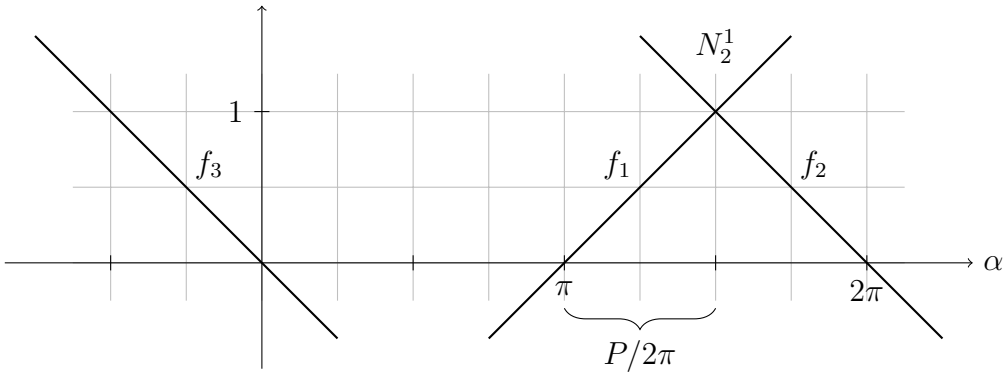


Abbildung 5.5: Beweisidee zur „kreisförmigen“ Dreiecksfunktion für B-Spline-Funktionen mit $K = 1$ und $P = 4$ als Darstellung über eine Minimum-Maximumfunktion auf drei Geraden, hier illustriert am Beispiel N_2^1.

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-

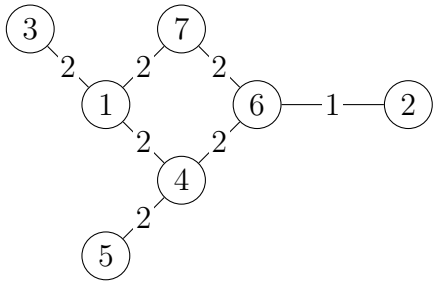
mit die Dreiecksfunktion, indem sie sich stets für das Minimum von $\max(f_1, 0)$ bzw. $\max(f_2, 0)$ entscheidet. Folglich beschreibt $\max(\max(f_3, 0), f_\triangle)$ die B-Spline-Funktion N_p^1 im Intervall $(0, 2\pi]$ für alle $p \in \{0, \ldots, P-1\}$.

5.5 Pooling auf Graphen

Neben den Faltungsschichten gehören die sogenannten Poolingschichten zu den fundamentalen Schichten eines CNNs. Poolingschichten 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 [24].
Eine Pooling-Operation auf Graphen erfordert dafür analog zum Pooling auf einem regulären Gitter eine logische Zusammenfassung von Knoten zu Clustern [6]. Der zu verwendende Clusteralgorithmus hat dabei jedoch einige Anforderungen, um als Poolingoperation in einem Netz genutzt werden zu könnnen. So muss dieser vorallem 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 [6]. 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 Poolingschichten im Netz. Es ist weiterhin notwendig, dass die Poolingoperation benutzerspezifische Poolinggrößen ermöglicht. Hier erscheinen vorallem Clustertechniken sinnvoll, die die Größe eines Graphen um den Faktor zwei reduzieren [6]. Damit können größere Poolinggrößen über die mehrfache Anwendung des Clusteralgorithmus realisiert werden.

5.5.1 Graphvergröberung

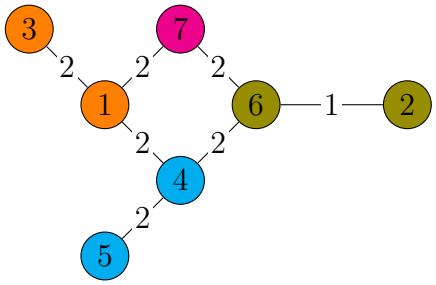
Es existieren eine Reihe von Clustertechniken auf Graphen [6, 7, 23]. Defferrard u. a. benutzen für die Poolingschicht eines Netzes auf Graphen die *Vergröberungsphase* des mehrstufigen Clusteralgorithmus *Graclus* [7]. Graclus zeigt sich dabei als besonders erfolgreich über einer Vielzahl von Graphen bei minimaler Berechnungskomplexität [6]. 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 [7]. 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 [7].



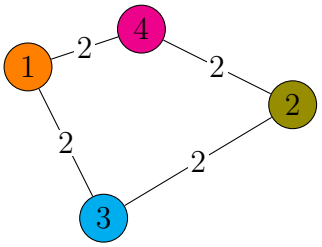
(a) Graph mit zufälliger Ordnung

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

Die Kantengewichte $\mathcal{E}_{l+1}$ werden dann über die Summe der Kantengewichte der jeweiligen Knotenuntermengen ohne Schleifen gebildet [7].

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 [7]. Bei weiteren Anwendungen des Clusteralgorithmus werden die Knoten im Folgenden anhand ihrer

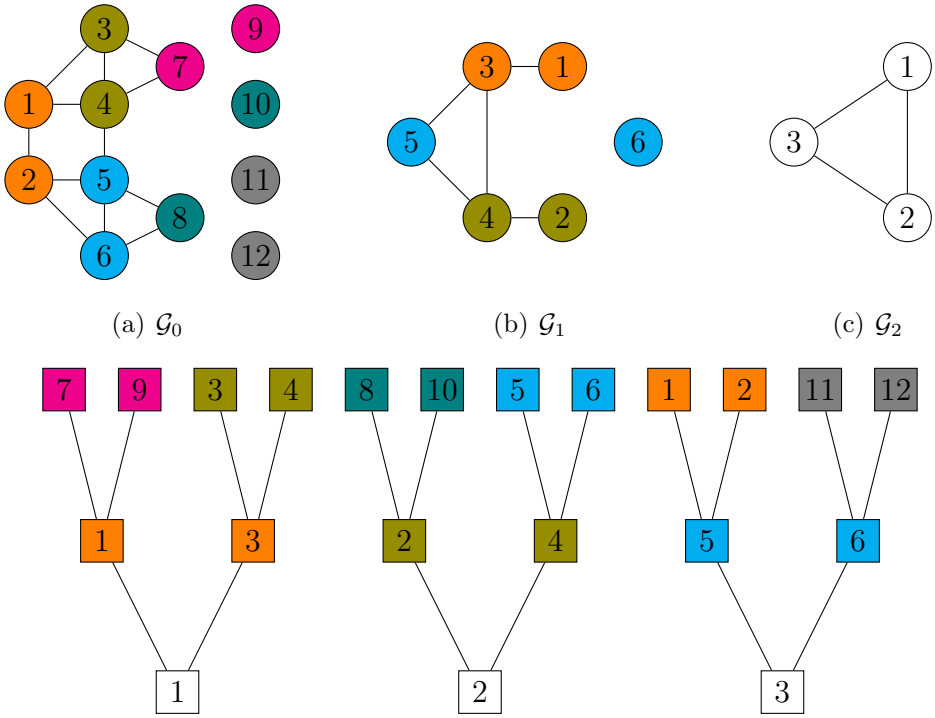
aufsteigenden Knotengrade durchlaufen [6].
Strategien für die Nachbarschaftsauswahl basieren üblicherweise auf der Maximierung von $w(v_i, v_j)$ [7]. Eine darauf aufbauende Strategie ist der *Normalized-Cut*

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

der die Kantengewichte relativ zu ihrem Knotengrad gewichtet [6, 7]. 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 [6]. 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 Poolingschichten. 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 Poolingoperation 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 [6]. Es ist jedoch möglich, die Knoten des Graphen so anzuordnen, dass eine Poolingoperation auf diesen in linearer Zeit, d.h. $\mathcal{O}(N)$, realisiert werden kann [6].
Nach jedem Vergröberungsschritt besitzt ein Knoten entweder genau einen oder zwei Kinder, je nachdem 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 *Fakeknoten* ohne Kanten in den Graphen $\mathcal{G}_l$ hinzugefügt [6]. 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 binären Bäumen

Abbildung 5.7: Illustration einer Poolingoperationen der Größe 4 (bzw. zweier Poolingoperationen 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 Fakeknoten zu $\mathcal{G}_1$ (1 Knoten) und $\mathcal{G}_0$ (4 Knoten) hinzugefügt werden. Mit der Anordnung der Knoten zu binären Bäumen (d) kann die Poolingoperation $\max(\cdot)$ 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 Fakeknoten $\mathbf{f}_2, \mathbf{f}_6, \mathbf{f}_{11}, \mathbf{f}_{12}$ 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 Fakeknoten als Kinder und (c) Fakeknoten besitzen immer genau zwei Fakeknoten als Kinder [6]. Abbildung 5.7 illustriert die Konstruktion eines solchen Baumes. Die Merkmale an den Fakeknoten 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 Poolingoperation [6]. Ebenso kann auf diesen Fakeknoten 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 Poolingoperation völlig analog zu einem eindimensionalen Gitter mit einer Größe und Schrittweise von zwei definieren können [6]. Diese Anordnung der Knoten macht den Prozess des Poolings extrem effizient und erlaubt ebenso eine parallele Verarbeitungsarchikektur auf GPUs [6].

Größere Poolinggrößen 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 euklidischen 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 euklidischen Raum.

5.6 Netzarchitektur

Die Architektur eines neuronalen Netzes auf Graphen mit spektralen Faltungen verhält sich durch die ähnliche Formulierung einer Faltungs- und einer Poolingschicht analog zu der Netzarchitektur klassischer CNNs. Dabei werden wie gewohnt mehrere Faltungsschichten mit stetig erhöhter Merkmalsausbreitung aneinander gereiht und an einigen Stellen über Poolingschichten getrennt, die die Anzahl der zu betrachtenden Knoten sukzessive reduzieren. Im Anschluss darauf finden sich im Allgemeinen zwei bis drei vollverbundene Schichten, die die Merkmalsgröße dann schlussendlich auf die gewünschte Ausgabegröße reduzieren [24]. Die mehrmalige Verkettung von Faltungsschichten sorgt dafür, dass auch bei einer relativ kleinen Faltung über die lokale Nachbarschaft eines jeden Knoten Merkmale weit entfernterer Knoten gewonnen werden können.
Ein CNN auf Bildern erfordert dabei in einer analogen Netzarchitektur eine feste Eingabegröße. Dafür werden die Bildermengen in der Regel skaliert und zugeschnitten, so dass diese alle die gleiche Bildgröße besitzen (zum Beispiel 224×224) [14]. Es erscheint jedoch schwierig, eine Menge von Graphen insofern anzupassen, dass diese alle die gleiche Anzahl an Knoten aufweisen. So ist es zwar vorstellbar, zusätzliche Fakeknoten zu jedem Graphen hinzufügen, damit diese alle eine feste Anzahl an Knoten aufweisen. Neben dem erhöhtem Speicheraufwand ist dieser Ansatz jedoch insbesondere nicht geeignet für unbekannte Graphen, die in das Netz eingespeist werden. So können diese eventuell eine größere Anzahl als die zuvor festgelegte Größe aufweisen. Ebenso liefert uns der Prozess der Graphvergröberung eine stets unterschiedliche Repräsentation eines Graphen, die wohlmöglich bereits eine größere Menge an Fakeknoten hinzufügt und dabei die festgelegte Knotengröße der Graphen überschreitet. Es ist weiterhin schwierig, einen Graphen auf eine feste Größe zuzuschneiden. So ist es insbesondere nicht ersichtlich, welche Knoten aus einem Graphen entfernt oder zusammengefasst werden können.
Die Architektur eines neuronalen Netzes auf Graphen erfordert folglich eine Struktur, die auf dynamischen Eingabegrößen operiert. Dazu muss jedoch zuerst geklärt werden, warum ein klassisches CNN auf Bildern eine feste Eingabegröße erfordert, denn die Faltungsschichten eines CNNs können Merkmalskarten beliebiger Größe generieren [14]. Die vollverbundenen Schichten jedoch brauchen rein nach ihrer Definition eine feste Eingabegröße [14]. Dadurch ergibt sich die Bedingung einer festen Eingabegröße lediglich durch den Übergang von einer Faltungs- zu einer vollverbundenen Schicht [14]. Eine einfache und elegante Lösung zur Umgehung dieses Problems bietet uns eine weitere Schicht, die zwischen der Faltungs- und vollver-

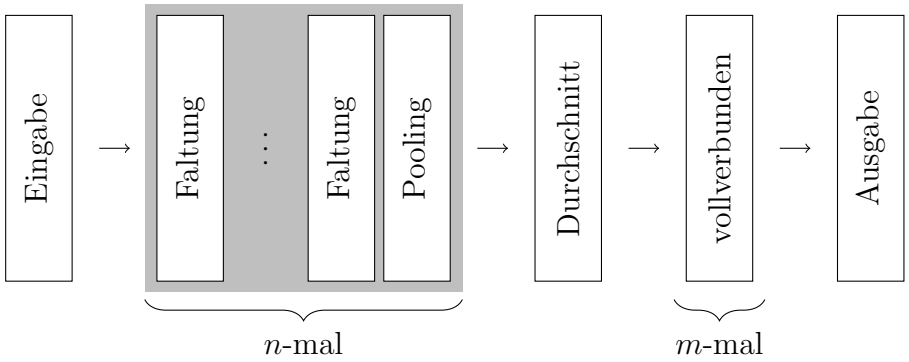


Abbildung 5.8: Typische spektrale Netzarchitektur auf Graphen bestehend aus beliebig vielen verketteten Faltungsschichten gefolgt von jeweils einer Poolingschicht. Im Anschluss sorgt die Benutzung einer Durchschnittsbildung auf den Knoten jedes Merkmals für die Verwendung von vollverbundenen Schichten hinzu zur Ausgabe.

bundenen Schicht des Netzes hängt und dafür sorgt, die Ausgabekarten der letzten Faltungsschicht auf eine feste Größe zu projizieren. Die Operation, die dafür üblicherweise genutzt wird, ist die Durchschnittsbildung eines Merkmals über all seinen Knoten. Da die Anzahl der betrachteten Merkmale pro Knoten in jeder Schicht fest ist, liefert uns die Durchschnittsbildung zwischen der Faltungs- und der vollverbundenen Schicht eines Netzes stets eine feste Anzahl an zu betrachtenden Merkmalen. Es ist jedoch anzumerken, dass diese Operation auf Bildern insbesondere translationsinvariant ist und folglich die Position der Merkmale im Bild verloren gehen. Graphen, kodiert als Adjazenzmatrizen $\mathbf{A}_{\text{dist}}$ und $\mathbf{A}_{\text{rad}}$, sind jedoch bereits translationsinvariant und folglich gehen dabei keine Informationen verloren. Es ist jedoch darauf zu achten, dass die Anzahl der Knoten in der letzten Faltungsschicht relativ klein und Merkmale auf den Knoten folglich bereits den globalen Graphen abdecken sollten.

Abbildung 5.8 veranschaulicht die beschriebene typische spektrale Netzarchitektur. Alternative Architekturen bzw. alternative Ansätze im Hinblick auf die Durchschnittsbildung zwischen der Faltungs- und der vollverbundenen Schicht werden in Kapitel 7 diskutiert.

6 Evaluation

6.1 Versuchsaufbau

6.1.1 Datensätze

MNIST. [22]

Cifar-10. [21]

Pascal VOC. [9]

6.1.2 Metriken

6.1.3 Parameterwahl

Vorstellung aller Parameter Superpixelalgorithmen Parameterwahl

Augmentierung von Graphen.

6.2 Merkmalsselektion

6.3 Augmentierung von Graphen

6.4 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 aufgrund der kleineren Anzahl an Trainingsparametern einzuschränken.

6.5 Laufzeitanalyse

Vergleich mit anderen Implementierungen.

6.6 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

Algorithmenverzeichnis

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] Biggs, Norman L.; ; Lloyd, E. Keith; Wilson, Robin J.: *Graph Theory.* Oxford University Press, 1999

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

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

[6] 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

[7] 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

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

[9] 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

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

[11] Fulkerson, Brian; Vedaldi, Andrea; Soatto, Stefano: Class Segmentation and Object Localization with Superpixel Neighborhoods. In: *International Conference on Computer Vision* (2009), S. 670–677

[12] Gadde, Raghedeep; Jampani, Varun; Kiefel, Martin; Kappler, Daniel; Gehler, Peter V.: Superpixel Convolutional Networks using Bilateral Inceptions. In: *European Conference on Computer Vision* (2016)

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

[14] He, Kaiming; Zhang, Xiangyu; Ren, Shaoqing; Sun, Jian: Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2014), S. 346–361

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

[16] He, Shengfeng; Lau, Rynson W. H.; Liu, Wenxi; Huang, Zhe; Yang, Qingxiong: SuperCNN: A Superpixelwise Convolutional Neural Network for Salient Object Detection. In: *International Journal of Computer Vision* (2015), S. 330–344

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

[18] Hu, Ming K.: Visual Pattern Recognition by Moment Invariants. In: *IRE Transactions on Information Theory* (1962), S. 179–187

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

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

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

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

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

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

[25] Organick, Elliott I.: *A Fortran IV Primer*. Addison-Wesley, 1966

[26] 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

[27] Saad, Yousef: *Iterative Methods for Sparse Linear Systems*. Society for Industrial and Applied Mathematics, 2003

[28] 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

[29] Siedhoff, Dominic: *A Parameter-Optimizing Model-Based Approach to the Analysis of Low-SNR Image Sequences for Biological Virus Detection*, Technische Universität Dortmund, Dissertation, 2016

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

[31] Stackoverflow: *Determine adjacent regions in numpy array*. `https://stackoverflow.com/questions/38073433/determine-adjacent-regions-in-numpy-array`. 2016

[32] Stiene, Stefan: *Konturbasierte Objekterkennung aus Tiefenbildern eines 3D-Laserscanners*, Universität Osnabrück, Diplomarbeit, 2015

[33] Stutz, David: *Superpixel Segmentation using Depth Information*. http://davidstutz.de/. 2014

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

[35] 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