Vorwort: Für alle zu implementierenden Aufgaben werden entsprechende Demonstrationsmöglichkeiten erwartet.

1 Bildfaltung

1.1 Was versteht man unter Bildfaltung?

Die Bildfaltung ist eine Operation die ein Bild mithilfe eines Filters manipuliert. Sie wird sowohl im Bereich des maschinellen Lernens bei Convolutional Neural Networks als auch im Bereich der traditionellen Bildverarbeitung eingesetzt. Einfachere Bildfaltungen sind zum Beispiel: Weichzeichnen (engl. blur), Schärfen (engl. sharpen) oder auch die Kantendetektion (engl. edge detection). Die Grundidee dabei ist, den Wert eines jeden Pixels auf Basis der Werte seiner Nachbarpixel als auch auf Basis seines eigenen Wertes zu berechen. Dabei wird jeder Nachbarpixel, sowie der eigene Wert zuerst mit einem vorher definierten Faktor multipliziert und dann alle so erhaltenen Werte aufsummiert. Diese Summe wird als neuer Wert für den betrachteten Pixel angenommen. Wie viele Nachbarn betrachtet werden und mit welchen Faktoren sie multipliziert werden wird durch eine Schablone bestimmt. Wie man sich so eine Schablone vorstellen kann ist in Abbildung 1 dargestellt. Legt man die Mitte der Schablone auf einen Pixel, so zeigt sie welche Pixel betrachtet werden und mit welchen Faktoren die jeweiligen Pixel multipliziert werden sollen. Diese Schablone wird nun Pixel für Pixel verschoben und der errechnete Wert in ein neues Bild gespeichert. Die Schablone wird meist als Kernel oder auch Filterkernel oder nur Filter bezeichnet.

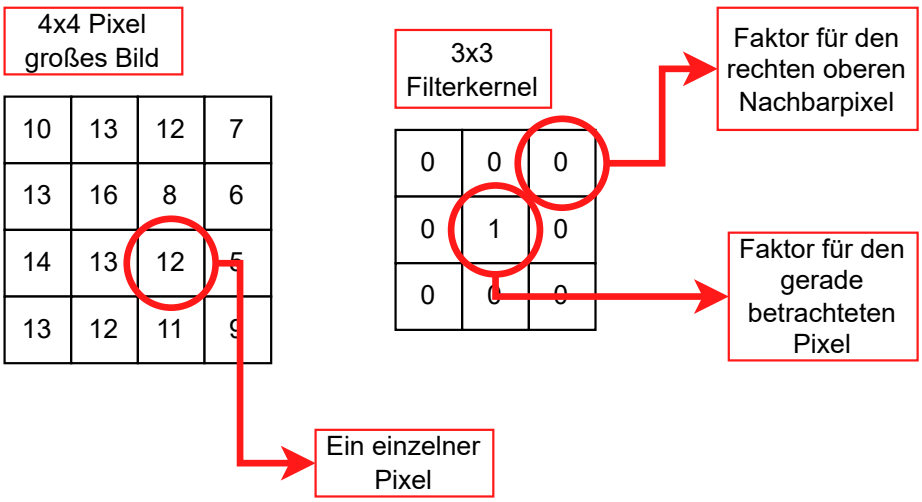

Figure 1: Terminologie bei der Bildfaltung.

Zur Vereinfachung gehen wir in diesem Arbeitsblatt immer von Graustufen-

bildern aus. Das gleiche Prinzip lässt sich aber auch auf Farbbilder anwenden

1.2 Definieren des Bildes

Eine Frage die man sich dabei stellen muss, ist, wie mit Positionen der Schablone umgegangen wird, für die kein Pixelwert im Bild existiert. Legt man die Mitte eines 3×3 Filterkernels bei dem Pixel (0, 0) an, so ist die erste Zeile und erste Spalte des Kernels ausserhalb des Bildes. Hierfür existieren mehrere Lösungsmöglichkeiten und die Entscheidung für eine Variante hängt vom Anwendungsfall ab. Die Einfachste Möglichkeit ist, Werte ausserhalb des Bildes mit dem Wert 0 zu belegen (auch zero-padding genannt). Eine Weitere Möglichkeit ist es, den Rand des Bildes nach aussen hin zu duplizieren (auch clamping genannt). Beide Möglichkeiten werden in Abbildung 2 vorgestellt.

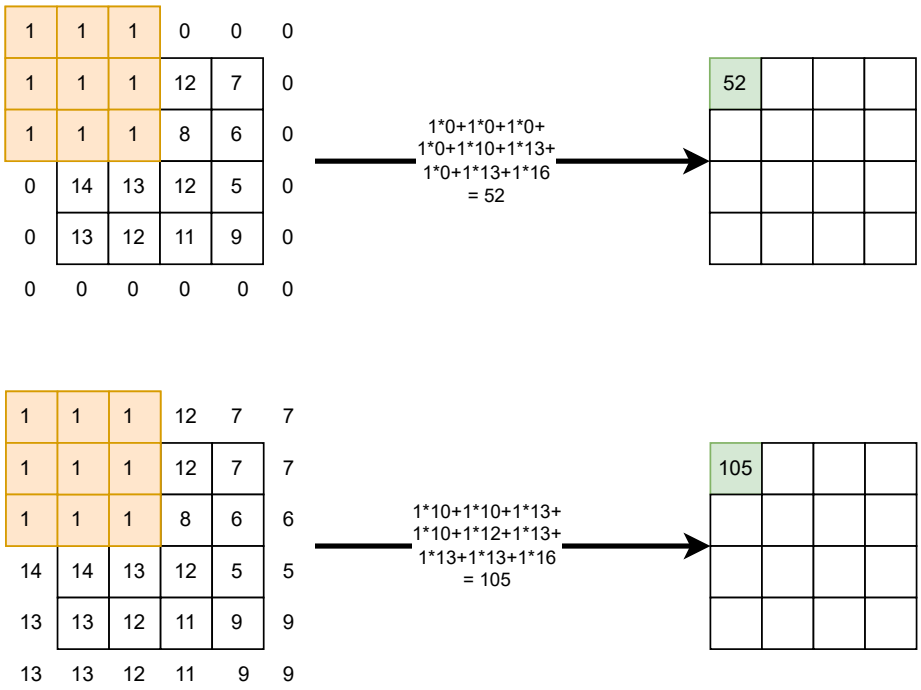

Figure 2: Verschiedene Arten den Rand eines Bildes zu falten. Oben: Zero-padding, Unten: Clamping.

2 Verschiedene Filterkernel

Es gibt viele verschiedene Filterkernel, doch für dieses Arbeitsblatt konzentrieren wir uns auf den Prewitt-Filter und den Boxfilter. Das nette an den meisten

Filtern ist, dass man aus dem Aussehen der Matrix bereits darauf schließen kann, wie ein Filter ein Bild verändert.

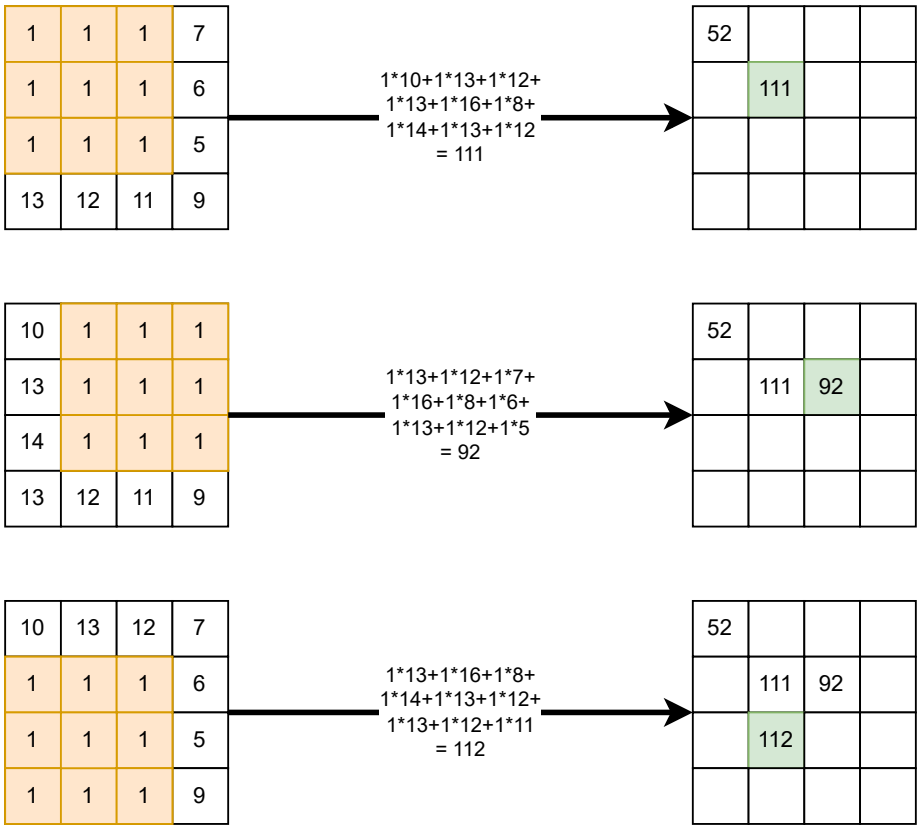

Figure 3: Faltung innerhalb eines Bildes (mit zero-padding).

2.1 Prewitt-Filter

Den Prewitt-Filter gibt es, wie in Abbildung 4 dargestellt, in einer horizontalen und einer vertikalen Ausführung. Intuitiv lässt sich sagen, dass dieser Filter die Nachbarpixel miteinander verrechnet. Da eine Reihe jeweils negativ ist, lassen sich so Änderungen in den Pixelwerten verstärken. Sind die Pixel links und rechts, bzw. oben und unten, nicht sonderlich verschieden, so ist auch die Summe des Filters ungefähr 0. Unterscheiden sich die Nachbarpixel stark voneinander, so wird auch die Summe größer oder kleiner. Man spricht hier auch von einer Kantendetektion.

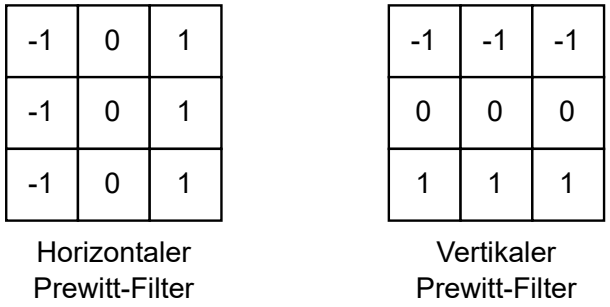

Figure 4: Der horizontale (links) und der vertikale (rechts) PrewittFilter.

2.2 Boxfilter

Beim Boxfilter ist jedes Filterelement definiert als 1/AnzahlDerElemente. Zur einfacheren Darstellung wird der Filter oft als Produkt eines Skalars und einer Einsmatrix notiert. Beide Varianten sind in Abbildung 5 dargestellt. Dieser Filter summiert alle Pixelwerte auf und teilt sie durch deren Anzahl, er bildet also den Mittelwert. Diese Mittelwertbildung ist eine Einfache Methode, um ein Bild weichzuzeichnen. Diesen Filter kann man beliebig groß definiert und er wird of auch als Averaging-Filter bezeichnet.

3 Umsetzung im Computer

Etwas technischer ausgedrückt, wird sowohl das Bild als auch der Filterkernel mittels einer 2D-Matrix im Computer dargestellt. Nun wird für jedes Element der Bildmatrix (i, j) die Filtermatrix durchlaufen, und zwar so, dass das mittlere Element der Filtermatrix an der Stelle (i, j) anliegt. Nun wird jeder Faktor der Filtermatrix mit dem entsprechenden Element der Bildmatrix multipliziert und die Ergebnisse aufsummiert. Ein Auszug dieses Vorgehens ist in Auflistung 1 als Pythoncode dargestellt. Diese Summe wird in eine neue Matrix an der Stelle (i, j) eingetragen. Am Ende erhält man ein neues, fertig gefiltertes Bild, wie in Abbildung 6 dargestellt.

```
methode get_filtered_pixel(i, j, image, kernel, behavior):
 kernel_half = int((len(kernel) - 1) / 2)
 final_pixel_value = 0
for k in range(-kernel_half, kernel_half + 1):
for l in range(-kernel_half, kernel_half + 1):
 kernel_factor = kernel[k+kernel_half][l+kernel_half]
 current_pixel = behavior.get_pixel_value(i+k, j+l, image)
 final_pixel_value += current_pixel * kernel_factor
return final_pixel_value
```

3x3 Boxfilter

$$\frac{1}{9} * \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \end{bmatrix}$$

5x5 Boxfilter

$$\frac{1}{25} * \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 1/25 & 1/25 & 1/25 & 1/25 & 1/25 \\ 1/25 & 1/25 & 1/25 & 1/25 & 1/25 \\ 1/25 & 1/25 & 1/25 & 1/25 & 1/25 \\ 1/25 & 1/25 & 1/25 & 1/25 & 1/25 \\ 1/25 & 1/25 & 1/25 & 1/25 & 1/25 \end{bmatrix}$$

Figure 5: Der Boxfilter einmal als Produkt (links) und einmal ausmultipliziert (rechts). Dargestellt ist ein 3×3 Filter und ein 5×5 Filter.

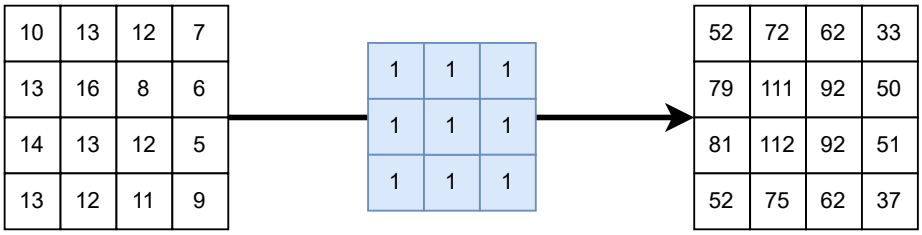

Figure 6: Das fertig gefilterte Bild (mit zero-padding).

3.1 Das pgm-Bildformat

Das pgm-Bildformat ist ein textbasiertes Bildformat für Graustufenbilder welches sich besonders eignet, wenn man aus einfachen Pixelwerten ein Bild generieren will. Es ist sozusagen das csv-Format der Bilder. Diese Bilder lassen sich mit GIMP, Krita oder Irfranview öffnen. Ein pgm-Bild hat folgende Einträge, die jeweils durch ein Leerzeichen oder eine neue Zeile getrennt werden:

- eine Magicnumber: P2 (was bedeutet diese Nummer?)
- eine Breite und eine Höhe in Pixeln
- den maximalen Wert den ein Pixel annehmen kann
- Eine Liste der einzelnen Pixelwerte (meist ein Pixel pro Zeile)
- Zeilen, die mit # beginnen, sind Kommentare und werden ignoriert

Listing 2:

```
P2
# A comment
4 4
255
52 72 62 33
79 111 92 50
81 112 92 51
52 75 62 37
```

Listing 3:

```
P2
# A comment
4 4
255
52
72
62
33
79
111
92
50
...
```

Listing 2 und Listing 3 zeigen, wie ein pgm-Bild mit dem gefalteten Resultat aus Abbildung 6 aussehen würde. Wie das Bild dann in einem Grafikprogramm wie etwa GIMP aussieht, zeigt Abbildung 7.

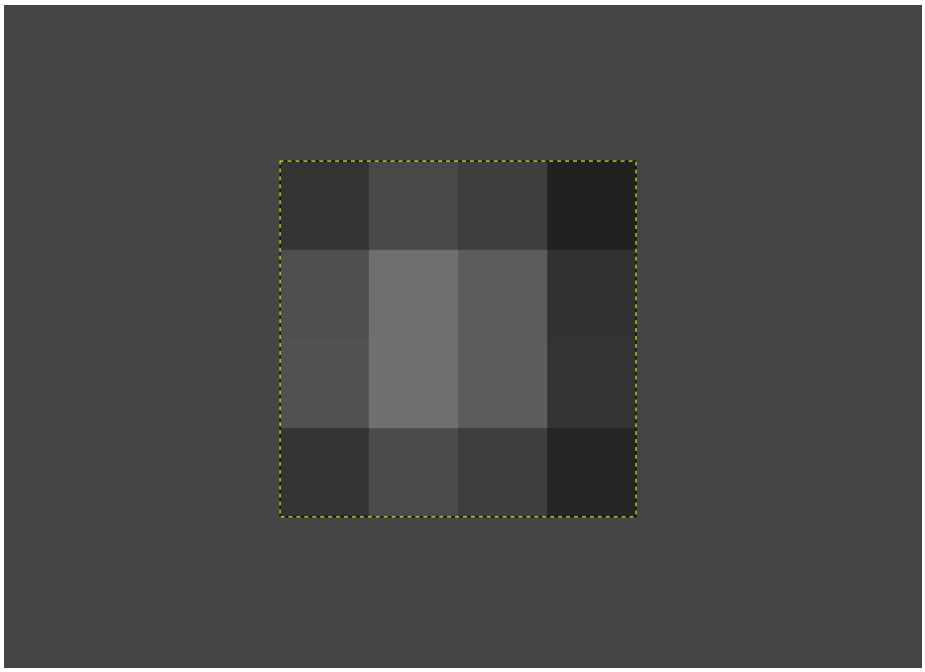

Figure 7: Das pgm-Bild aus Auflistung 2, wie es in GIMP bei 9000- fachem Zoom erscheint. Der hohe Zoomfaktor ist nötig da das Bild nur 4×4 Pixel groß ist!

4 Die Aufgabe

1. Implementieren Sie eine Klasse Image (1p), die folgende Methoden bietet:

    1. readFromFile(filename) um ein pgm-Bild in ein zweidimensionales int-Array einzulesen. Ihr müsst mindestens die Variante aus Listing 3 lesen können, d.h. eine Zahl pro Zeile. Die Auflösung des Bilds ist bei 1500x1500 begrenzt. Entspricht das Bild nicht einem der beiden vorgestellten Schemata, soll ein Fehler ausgegeben werden (2p).

    2. writeToFile(filename) um ein zweidimensionales int-Array in ein pgm-Bild zu speichern. Hier bietet sich an, das Schema aus Listing 2 zu nehmen, da dies einfacher als Textdatei zu lesen ist (3p).

    3. convolve(kernel, borderBehavior) um das Bild mit einen Filterkernel zu falten. Das Verhalten am Rand des Bildes wird dabei durch das Objekt borderBehavior festgelegt. Das Ergebnis der Faltung ist ein neues Bild das als Rückgabewert übergeben wird. Da wir die Mitte des Filterkernels ermitteln müssen, muss ein Filterkernel immer eine ungerade Anzahl an Spalten und Zeilen haben! (5p)

2. Eine abstrakte Klasse BorderBehavior mit einer abstrakten Methode getPixelValue(i, j, image) (1p), die den Pixelwert des übergebenen Bildes image an der Stelle (i, j) zurückgibt. Von dieser Klasse werden zwei Spezialisierungen abgeleitet:

    1. ZeroPaddingBorderBehavior, welche für (i, j) ausserhalb des Bildes, wie in Abbildung 2 oben dargestellt, den Wert 0 zurückgibt (2p).

    2. ClampingBorderBehavior, welche für (i, j) ausserhalb des Bildes, wie in Abbildung 2 unten dargestellt, den Wert des am nächsten gelegenen Randpixels zurückgibt (2p).

3. Eine Klasse KernelFactory (1p), die folgende statische Methoden bietet:

    1. createVerticalPrewittKernel um einen vertikalen 3×3 PrewittFilter zu erstellen. (1p)

    2. createHorizontalPrewittKernel um einen horizontalen 3×3 Prewitt-Filter zu erstellen. (1p)

    3. createBoxFilter(size) um einen quadratischen Box-Filter der größe size zu erstellen. Bei einem Box-Filter hat jedes Element den Wert 1/AnzahlDerElemente. Bei einem 3×3 Box-Filter hat also jedes Element den Wert 1/9 (3p).

4. Implementieren Sie eine Main-Methode, die folgende Filteroperationen einmal mit zeropadding und einmal mit clamping durchführt und das Ergebnis jeweils in einem Ordner ZeroPadded und Clamped speichert. Insgesamt erhält man so 10 gefilterte Bilder. (1p)

1. Das gegebene Bild1.pgm (Benutzen Sie GIMP für die Erstellung der Testbilder) wird je einmal den horizontalen und einmal mit dem vertikalen Prewitt-Filter gefiltert und das Ergebnis als pgm-Bilder in die Dateien Ergebnis 1-horizontal.pgm und Ergebnis 1-vertical.pgm abspeichert. Wieso liefert hier zeropadding ein minimal anderes Ergebnis als clamping? (1p)

2. Das gegebene Bild2.pgm wird je einmal mit einem Box-Filter der Größe 3, 11 und 27 gefiltert und die Ergebnisse jeweils in die Dateien Ergebnis 2-3.pgm, Ergebnis 2-11.pgm und Ergebnis 2-27.pgm speichert. Was bewirkt den Ergebnissen nach zu urteilen eine Vergrößerung des Filterkernels bei einem Boxfilter? Wie wirken sich zero-padding und clamping hier aus? (1p)

3. Geben Sie sowohl die eingelesene Bilder (Bild1.pgm und Bild2.pgm) als auch alle Ergebnisse der Faltung auf dem Bildschirm aus. (1p)

5. Erstellen Sie eine geeignete Dokumentation des Programms in UML mit einer textuellen Beschreibung der Funktionsweise der Lösung. (7p)