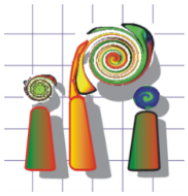

Institut für Photogrammetrie und GeoInformation
Leibniz Universität Hannover

Masterarbeit

Überwachte Detektion von Bombenkratern in historischen Luftbildern mittels Convolutional Neural Networks

Dominic Clermont

Hannover, 04. März 2019

Erstprüfer: apl. Prof. Dr. techn. Franz Rottensteiner
Zweitprüfer: M.Sc. Christian Kruse

Institut für Photogrammetrie
und GeoInformation

Leibniz
Universität
Hannover

Institut für Photogrammetrie und GeoInformation,
Nienburger Straße 1, 30167 Hannover

Thema der Masterarbeit von Herrn B.Sc. Dominic Clermont

Überwachte Detektion von Bombenkratern in historischen Luftbildern mittels Convolutional Neural Networks (Arbeitstitel)

Künstliche neuronale Netze haben sich durch in den letzten Jahren als hervorragende Methode im Bereich des maschinellen Lernens erwiesen. Insbesondere bei der Objektdetektion in Bildern kommen dabei sogenannte Convolutional Neural Networks (CNNs) zum Einsatz. Diese sind dazu in der Lage, Objekte anhand selbstständig erlernter, abstrakter Merkmale zu detektieren. Um ein Netz auf die Erkennung von bzw. Unterscheidung zwischen verschiedenen Objekten zu trainieren, sind große Mengen an Trainingsdaten nötig, d.h. von Daten, für die das gewünschte Ergebnis bekannt ist. CNNs können unter anderem durch Modifikation ihrer Architektur oder die Auswahl verschiedener Optimierungsverfahren beim Training an die jeweilige Aufgabenstellung angepasst werden.

Ziel dieser Masterarbeit ist die Anwendung von CNNs für die Detektion von Bombenkratern in historischen Luftbildern. Diese Aufgabe gliedert sich in zwei Teile. Im ersten Schritt soll ein CNN so trainiert werden, dass es für ein gegebenes Bild feststellen kann, ob darin ein Bombenkrater dargestellt wird oder nicht. Dies erfordert unter anderem die Gestaltung der Netzarchitektur, die Auswahl geeigneter Verfahren der Optimierung sowie das Erstellen eines Trainingsdatensatzes. Im zweiten Schritt sollen in beliebigen, für das Netz bis dahin unbekannten Luftbildern Krater detektiert werden. Dazu gehört unter anderem die Auswahl geeigneter Verfahren zum Vorschlag von relevanten Bildausschnitten, damit das Netz nicht an jeder Bildposition evaluiert werden muss. Zusätzlich sind für die Ergebnisse des Detektors geeignete Darstellungsformen zu wählen, sodass die Ergebnisse zur Unterstützung bei der manuellen Luftbildauswertung verwendet werden können. Schließlich sind die Methoden und deren Parameter bezüglich der Vollständigkeit und Korrektheit der Ergebnisse zu optimieren.

Die Methode soll mit Hilfe von vorhandenen Bausteinen wie z.B. der Software Tensorflow in Python implementiert und anhand eines vom Kampfmittelbeseitigungsdienst Niedersachsen (KBD) zur Verfügung gestellten Testdatensatzes evaluiert werden. Dieser Datensatz umfasst mehrere historische gescannte Luftbilder, in denen vom KBD alle Bombenkrater markiert wurden. Dabei liegen sowohl Informationen zur Position als auch zur Größe der Krater vor. Aus diesen Daten wird zunächst ein repräsentativer Trainingsdatensatz erstellt. Zu diesem Zweck müssen aus den Daten Bilder extrahiert werden, deren Inhalt jeweils einer der beiden Klassen zugeordnet werden kann, wobei diese Zuordnung mit den Bildausschnitten in geeigneter Weise gespeichert wird. Für die eigentliche Evaluierung wird der Detektor auf Bildausschnitte angewandt, aus denen keine Trainingsdaten extrahiert wurden, und die Ergebnisse werden mit den bekannten Positionen von Bombenkratern verglichen, um in der Folge diverse Qualitätsindices ableiten zu können.

apl. Prof. Dr. techn. Franz Rottensteiner / M.Sc. Christian Kruse

Fakultät für Bauingenieurwesen und Geodäsie

Institut für Photogrammetrie und GeoInformation

Prof. Dr.-habil. Christian Heipke

apl. Prof. Dr. Franz Rottensteiner,
M.Sc. Christian Kruse

Tel. +49 511 762-19388
Fax +49 511 762-2483
E-Mail: kruse
@ipi.uni-hannover.de

06.09.2018

Besucheradresse:
Nienburger Straße 1
30167 Hannover
www.ipi.uni-hannover.de

Kurzfassung

Die Folgen der Kampfhandlungen des letzten Weltkrieges sind auch heute noch gegenwärtig. Zahlreiche Fliegerbomben sind nicht explodiert und liegen verborgen im Erdreich. Das automatische Aufspüren solcher Blindgänger kann durch die Detektion von Bombenkratern angegangen werden: Von den Kratern kann auf eine potentielle Belastung durch Blindgänger geschlossen werden. In dieser Arbeit wird eine Methodik zur automatischen Detektion von Bombenkratern in Kriegsluftbildern vorgestellt. Dabei werden zunächst Kraterkandidaten aus einem Luftbild durch einen Blob-Detektor extrahiert. Anhand gegebener Referenzkrater kann für jeden Kandidaten überprüft werden, ob dieser tatsächlich einen Krater darstellt oder nicht. Kandidaten aus verschiedenen Bildern werden zu drei Datensätzen zusammengefasst. Diese Datensätze werden dazu verwendet, Convolutional Neural Networks (CNNs) im Rahmen eines Zwei-Klassen-Klassifikationsproblems zu trainieren, zu validieren und zu testen. Für die CNNs werden unterschiedliche Architekturen untersucht und miteinander verglichen. Außerdem wird die das Lernverhalten bestimmende Verlustfunktion der CNNs an die vorliegende Problematik angepasst. Ein trainiertes CNN kann damit zur Klassifikation von Kraterkandidaten verwendet werden. Die Kombination aus Extraktion von Kraterkandidaten und Klassifikation dieser Kandidaten kann somit zur automatischen Detektion von Bombenkratern genutzt werden.

Abstract

The aftermath of the air strikes during the second world war is still present today. Numerous bombs dropped by planes did not explode and may still exist in the ground. Tracking down these duds can be tackled by detecting bomb craters. The existence of a dud can be inferred from the existence of a crater. This work proposes a method for the automatic detection of bomb craters in aerial wartime images. First of all, crater candidates are extracted from an image using a blob detector. Based on given crater references, for every candidate it is checked whether it, in fact, represents a crater or not. Candidates from various aerial images are combined into three datasets. These datasets are used to train, validate und test convolutional neural networks (CNNs) in the context of a two-class classification problem. For the CNNs, three architectures are examined and compared to each other. Furthermore, a loss function (controlling what the CNNs are learning) is adapted to the given task. The trained CNNs can be used for the classification of crater candidates. Finally, by combining the extraction of crater candidates and their classification, bomb craters can be detected automatically.

Eigenständigkeitserklärung

Ich erkläre hiermit an Eides statt, dass ich die vorliegende Arbeit ohne Hilfe Dritter und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe; die aus fremden Quellen direkt oder indirekt übernommenen Gedanken sind als solche kenntlich gemacht. Die Arbeit wurde bisher in gleicher oder ähnlicher Form in keiner anderen Prüfungsbehörde vorgelegt und auch noch nicht veröffentlicht.

Dominic Clermont

Inhaltsverzeichnis

KAPITEL 1

Motivation und Zielsetzung

1.1 Motivation

Obwohl die letzten Kampfhandlungen des zweiten Weltkriegs schon mehr als 70 Jahre zurückliegen, stellen ihre Nachwirkungen noch immer eine Gefahr für die Menschen in Europa dar. Sogenannte Kampfmittelaltlasten liegen teilweise überwuchert im Wald oder verborgen im Erdreich. Diese Altlasten sind nicht-detonierte Munition wie Granaten oder Bomben. Besonders gefährlich sind dabei Fliegerbomben. Diese besitzen nicht nur eine enorme Sprengkraft, sondern sind auch relativ häufig nicht detoniert. Nach den Erfahrungswerten der Sprengkommandos im zweiten Weltkrieg sind $10\% - 15\%$ der von den Alliierten abgeworfenen Fliegerbomben Blindgänger gewesen. Im Jahr 2013 wurde die Zahl der nicht-explodierten Bomben in Deutschland auf ca. 100.000 geschätzt [Ruoff, 2013]. Seit 1947 gab es ca. 20 Selbstdetonationen in Deutschland [gubd.de, 2018].
Zum Auffinden der Altlasten werden heute in erster Linie historische Luftbilder genutzt, die damals nach den Bombardements aufgenommen wurden. Ein Beispiel eines solchen Bildes ist in Abbildung 1.1 dargestellt. Diese Bilder werden für Niedersachsen vom Kampfmittelbeseitigungsdienst manuell nach Bombenkratern abgesucht, um so Gebiete

festzustellen, die potentiell durch Blindgänger belastet sein können. Da das manuelle Auswerten eines Luftbildes sehr zeitaufwändig ist, können nicht alle verfügbaren Luftbilder - allein für Niedersachsen stehen mehr als 100.000 Bilder zur Verfügung - systematisch nach Bombenkratern abgesucht werden. Tatsächlich ist es so, dass eine solche Auswertung nur bei Bedarf, z.B. bei einem Bauvorhaben, für ein sehr begrenztes Gebiet geschieht.

Abbildung 1.1: Luftbild vom 23.04.1944 im Maßstab 1:24.000

Die Darstellung zeigt beispielhaft ein Luftbild, wie es auch in dieser Arbeit verwendet wird. Das Bild wurde im Original auf Film aufgenommen. In der unteren rechten Ecke ist die Bildnummer, während in der oberen linken Ecke Verschmutzungen und Bildfehler zu erkennen sind.

Aus dieser Problematik heraus entsteht der Bedarf nach einem automatischen Verfahren, welches dazu in der Lage ist, Bombenkrater zuverlässig in historischen Luftbildern zu detektieren. Für diese Aufgabenstellung, also die Detektion von Objekten in Bildern, eignen sich sogenannte Convolutional Neural Networks, kurz CNNs. Diese sind in ihrem Aufbau und ihrer Funktion inspiriert durch Vorstellungen über Aufbau und Funktion von Gehirnzellen. CNNs ermöglichen durch die Verwendung abstrakter, selbst-erlernter Merkmale die Klassifikation lokaler Bildausschnitte. Durch den technischen Fortschritt der letzten Jahre haben sich CNNs zu der State-of-the-Art-Methode für Aufgaben im Bereich der Bildklassifikation entwickelt. In Kombination mit einem geeigneten Verfahren zur Auswahl von Kraterkandidaten kann ein Algorithmus realisiert werden, welcher dazu in der Lage ist, Bombenkrater in Luftbildern zu detektieren. Somit kann eine flächendeckende und vor allem automatische Auswertung aller verfügbaren Luftbilder ermöglicht werden.

1.2 Stand der Forschung

Um den Stand der Forschung bei der Krater-Detektion zu untersuchen, werden nicht nur Arbeiten betrachtet, die sich mit der Detektion von Bombenkratern beschäftigen. Es wird auch auf Arbeiten eingegangen, die sich mit der Detektion von Einschlagskratern, z.B. durch Meteoriten auf Himmelskörpern, befassen. Aufgrund der Nähe der beiden Themen zueinander ähneln sich auch die Ansätze zur Lösung der jeweiligen Problemstellungen: So werden meist zuerst Positionen oder Bereiche aus einem Bild extrahiert, an welchen sich ein Krater befinden könnte; diese Position werden als Krater-Kandidaten bezeichnet. Anschließend wird jeder dieser Kandidaten bewertet oder klassifiziert, um festzustellen, ob es sich bei einem Kandidaten wirklich um einen Krater handelt. Im Folgenden werden beide Arbeitsschritte bei verwandten Arbeiten näher untersucht.

Eine Möglichkeit, Krater-Kandidaten aus einem Bild zu extrahieren, ist das Sliding-Window-Verfahren. Dabei wird ein Suchfenster von fester Größe in gleichmäßigen Abständen über das Bild geschoben, wobei jede Position des Suchfensters einen Krater-Kandidaten darstellt. So besteht bei [Merler et al., 2005] die Aufgabe darin,

Bombenkrater in historischen Luftbildern zu detektieren; die Luftbilder werden auf eine einheitliche Bodenpixelgröße skaliert, wodurch die Suchfenster in den Bildern eine gleiche Größe in Meter sowie in Pixel haben. Das Suchfenster wird dann um jeweils einen Pixel weitergeschoben. An jeder Position des Suchfensters wird aus dem Bildausschnitt im Suchfenster ein Krater-Kandidat gebildet. Einen sehr ähnlichen Ansatz verfolgen [Brenner et al., 2018]. Hier wird das Sliding-Window-Verfahren so variiert, dass das Suchfenster je um ein Viertel seiner Größe weitergeschoben wird. Hierdurch wird die Anzahl aller Krater-Kandidaten deutlich reduziert, allerdings geht auch die pixelbezogene Redundanz verloren, die für beide Arbeiten wichtig ist:
Zur Klassifikation der Krater-Kandidaten verwenden [Merler et al., 2005] eine Variante von AdaBoost, wohingegen [Brenner et al., 2018] die Kandidaten durch CNNs klassifizieren. In beiden Arbeiten wird auf diese Weise für jedes Pixel im Bild eine Wahrscheinlichkeit bestimmt, mit welcher sich an der Position des Pixels ein Krater befindet. Da Krater meist durch mehr als ein einzelnes Pixel dargestellt werden, müssen die klassifizierten Pixel nachträglich zu Anhäufungen (Clustern) vereinigt werden, um die Detektion von einzelnen Kratern zu ermöglichen. Dieser letzte Schritt könnte vermieden werden, indem nicht flächenhaft jedes Pixel klassifiziert wird. Stattdessen könnten die gebildeten Ausschnitte der Suchfenster als umgebende Rechtecke (Bounding-Boxes) einer Detektion interpretiert und klassifiziert werden.

Eine andere Möglichkeit zur Extraktion von Krater-Kandidaten besteht in der Verwendung des Kanade-Lucas-Tomasi-Detektors. Dieser wird in [Meng et al., 2013] genutzt, um Kandidaten für Einschlagskrater von Meteoriten auf Himmelskörpern zu extrahieren. Der Aufgabenbereich der Detektion von Einschlagskratern auf Himmelskörpern bietet die Möglichkeit, Vorwissen über die Licht- und Schattenverläufe innerhalb von Kratern zu nutzen. Zudem wird der Detektor, welcher lokale Unterschiede der Grauwerte der Pixel untersucht, nicht von irdischen Objekten, wie z.B. Häusern oder Bäumen, beeinflusst. Die Schattenwürfe dieser Objekte ähneln mit ihrer relativ dunklen, kompakten Erscheinungsform den zu detektierenden Bombenkratern. Damit ist dieser Ansatz vermutlich nicht für das vorliegende Problem geeignet. Eine ähnliche Problematik stellt sich bei der in [Urbach & Stepinksi, 2009] und [Cohen et al., 2016] verwendeten Methode dar: Hier werden die klaren Licht- und Schattenverläufe innerhalb von planetaren

Einschlagskratern genutzt, um Krater-Kandidaten zu erhalten. Zudem werden in dieser Variante viele an die verwendeten Daten angepasste Parametereinstellungen verwendet, welche vermutlich nicht für die vorliegende Arbeit übernommen und nur aufwändig angepasst werden können.
Eine weitere Möglichkeit zur Extraktion von Krater-Kandidaten wird in [Kruse et al., 2018] verwendet. Hier besteht ein Teilproblem in der Detektion von Bombenkratern in historischen Luftbildern, welches durch das Verfahren der markierten Punktprozesse angegangen wird. Dabei wird zur Optimierung des Verfahrens in der Vorverarbeitung ein Blob-Detektor verwendet, welcher sich in dieser Form auch als Methode zur Auswahl von Krater-Kandidaten eignet. Die Vermutung besteht darin, dass Krater in Bildern eine rundliche Erscheinung haben und damit durch einen Blob-Detektor gefunden werden können. Eine Einschränkung stellt in dieser Arbeit allerdings die Methodik dar, mit welcher die Krater-Kandidaten evaluiert werden; hierbei ist es nicht möglich, helle Krater zu detektieren. Solche Krater zeichnen sich dadurch aus, dass der innere Bereich heller erscheint als die Umgebung. Sie können dadurch zustande kommen, dass die Krater nach einem Bombardement wieder zugeschüttet wurden. Helle Krater kommen zwar vergleichsweise selten in Luftbildern vor, können aber durch dieses Verfahren nicht detektiert werden.

Für die Evaluierung bzw. Klassifikation jedes Krater-Kandidaten können unterschiedliche Methoden verwendet werden. In [Kruse et al., 2018] wird jeder Kandidat durch eine Energiefunktion bewertet, welche unter anderem auf lokalen Gradienten beruht. Damit wird die in dieser Arbeit getroffene Annahme über die rundliche Form der Krater wieder aufgegriffen. Diese Methode setzt allerdings voraus, dass die Modell-Annahmen über Form und Aussehen, in diesem Fall eine Ellipse mit hohem Grauwertgradienten am Rand, für alle Krater zutreffen. Dadurch können Krater, die dieses Annahmen nicht erfüllen, nicht detektiert werden. Zudem können andere Objekte, wie Häuser oder Bäume oder deren Schattenwürfe, die diese Annahmen erfüllen, fälschlicherweise als Krater detektiert werden.
Alternativen dazu stellen statistische Methoden dar, welche aus Beispielen lernen, wie Krater von anderen Objekten zu unterscheiden sind. So wird in [Brenner et al., 2018] ein Convolutional Neural Network verwendet. Die Architektur dieses CNNs entspricht

der in [Huang et al., 2016] als *DenseNet* vorgestellten Architektur, welche bei der Klassifikation von Bildern verwendet wurde. Ein mögliches Problem bei diesem Ansatz stellt der Umstand dar, dass nur die Architektur dieses CNNs übernommen wurde, nicht aber die gelernten Gewichte des Netzes, welche sich aus dem Training in [Huang et al., 2016] ergeben. Das Übernehmen der Architektur ermöglicht aber noch keine brauchbare Klassifikation, da das Netz noch mit, meist sehr vielen, Trainingsdaten trainiert werden muss. Zudem wird in der Arbeit berichtet, dass sich durch die verwendete Methode bei der Krater-Extraktion (Sliding-Window, s.o.) folgender Umstand ergibt: Der Großteil der extrahierten Kandidaten stellt tatsächlich keine Krater dar. Entsprechend ist der Anteil der Kandidaten, die tatsächlich einen Krater darstellen, sehr gering. Das ungleiche Verhältnis hatte vor allem im Anwendungsfall stark negative Auswirkungen auf die Klassifikationsgenauigkeit dieser Methode.
Eine Möglichkeit, dieses Problem anzugehen, wird in [Merler et al., 2005] genannt. Als Klassifikationsmethode wird hier eine Variante von AdaBoost verwendet. Darin wird eine zusätzliche Gewichtung eingeführt, die den Einfluss von positiven und negativen Trainingsbeispielen auf das Lernen beeinflusst. Dabei ist ein positives Trainingsbeispiel ein Kraterkandidat, für welchen bekannt ist, dass er tatsächlich einen Krater darstellt; ein negatives Trainingsbeispiel stellt entsprechend keinen Krater dar. Durch die Einführung dieser Gewichtung konnte die Klassifikationsgenauigkeit verbessert werden.

Die Detektion von Kratern kann durch die Kombination von Kandidaten-Extraktion und Klassifikation realisiert werden. In [Brenner et al., 2018] wird an dieser Stelle das Problem der ungleichen Klassenverhältnisse dadurch angegangen, dass räumliche Informationen über die Verteilung der Krater genutzt werden. Dazu wird die Annahme getroffen, dass Bombenkrater aufgrund der Abwürfe immer in Anhäufungen auftreten. Alleinstehende Detektionen werden deshalb verworfen, da dies die Annahme verletzen würde. Durch diesen Ansatz können demnach viele Kandidaten aussortiert werden, die keinen Krater darstellen. Das Problem dabei ist, dass die Größe dieser Anhäufungen vorher bekannt sein muss. Ist dies nicht hinreichend bekannt, können einzelne Detektionen fälschlicherweise verworfen werden. Im Anschluss an diesen Filterschritt wird eine Non-Maximum Unterdrückung angewendet, um einander zu sehr überlappende Detektionen herauszufiltern.

Alternativ zur Detektion von einzelnen Kratern wird in [Kruse et al., 2018] eine Kerndichteschätzung angewendet, um aus den einzelnen Krater-Detektionen eine Belastungskarte zu generieren. In einer solchen Karte werden Bereiche im Bild flächenhaft als entweder belastet oder unbelastet markiert. Ein Gebiet gilt als belastet, wenn dort wahrscheinlich Blindgänger vorhanden sind. Blindgänger treten wahrscheinlich nur in den Gebieten auf, in denen auch Bomben explodiert sind. Damit werden die Gebiete um detektierte Bombenkrater herum als belastet markiert. Hier ist die Annahme, dass die Detektion einzelner Krater zweitrangig ist; im Vordergrund steht die Detektion von belasteten Gebieten. Der Vorteil dabei ist, dass nicht alle Krater detektiert werden müssen.

1.3 Zielsetzung und Aufbau der Arbeit

Das Ziel dieser Arbeit besteht in der Detektion von Bombenkratern in historischen Luftbildern. Für diese Aufgabe werden vom Kampfmittelbeseitigungsdienst Niedersachsen (KBD) Luftbilder zur Verfügung gestellt, in denen Bombenkrater manuell durch Experten markiert wurden. Dadurch steht für jedes Luftbild eine Menge von Referenzkratern zur Verfügung, bei welchen die Bildposition und der Radius bekannt ist. Zunächst gilt es, die Extraktion von Krater-Kandidaten aus den Bildern in geeigneter Weise zu realisieren. Anschließend sollen daraus in Zusammenspiel mit den gegebenen Daten Datensätze erstellt werden, welche für das Trainieren und Testen eines Convolutional Neural Networks verwendet werden. Das CNN soll die Krater-Kandidaten in zwei mögliche Klassen einteilen: Die Kandidaten stellen entweder Krater (positives Trainingsbeispiel) oder Hintergrund bzw. keine Krater (negatives Trainingsbeispiel) dar. Für die Klassifikation der Kandidaten werden unterschiedliche CNN-Architekturen untersucht. Zusätzlich wird für das Training der Netze die Verlustfunktion modifiziert. Vermutlich werden bei der Extraktion der Kandidaten nicht gleich viele Kandidaten extrahiert, die tatsächlich einen Krater darstellen, wie Kandidaten, die tatsächlich einen Krater darstellen. Dieses ungleiche Verhältnis wird beim Training durch Anpassung der Verlustfunktion berücksichtigt. Abschließend wird die Extraktion der Kraterkandidaten mit der Klassifikation durch die CNNs kombiniert, um die Detektion von Bombenkratern zu realisieren. Sowohl die Extraktion von Krater-Kandidaten als auch

die Klassifikation durch die CNNs und die Kombination von beidem werden anhand geeigneter Qualitätsmaße evaluiert und verglichen.

Die Arbeit gliedert sich wie folgt: In Kapitel 2 werden die für die entwickelten Methoden benötigten Grundlagen erklärt. Dabei wird vor allem auf Neuronale Netze und ihre Funktionsweise eingegangen. Danach folgt mit Kapitel 3 die Ausführung über die entwickelten Methoden in der Reihenfolge, die durch die Zielsetzung gegeben ist: Zuerst wird ein Verfahren zur Krater-Kandidaten-Auswahl in geeigneter Weise realisiert, um damit anschließend die Erstellung der Datensätze zu ermöglichen. Diese werden zum Trainieren, Validieren und Testen der CNNs verwendet. Darauf folgt die Entwicklung unterschiedlicher CNN-Architekturen sowie die Wahl geeigneter Lernverfahren. Das Kapitel schließt mit Ausführungen über die Detektion als Kombination von Krater-Kandidaten-Auswahl und Klassifikation ab. In Kapitel 4 werden Qualitätsmaße eingeführt, anhand derer die Methoden in geeigneter Weise evaluiert und verglichen werden. Zudem werden hier die verwendeten Daten beschrieben und im Rahmen einer Vorverarbeitung an die Aufgabenstellung angepasst. In Kapitel 5 folgt das Fazit über den in dieser Arbeit geleisteten Beitrag. Abschließend wird ein Ausblick auf Verbesserungsmöglichkeiten und mögliche weitere Arbeiten gegeben.

KAPITEL 2

Grundlagen

Im folgenden Kapitel werden die Grundlagen erklärt, die für die in dieser Arbeit entwickelten Methoden benötigt werden. Da der Fokus der Arbeit auf der Anwendung und Optimierung von neuronalen Netzen liegt, wird in diesem Kapitel das Augenmerk auf deren Funktionsweise gelegt.
In Abschnitt 2.1 wird der Grundbaustein neuronaler Netze, das Neuron, erklärt. Darauf folgend wird das Perzeptron als einfachstes neuronales Netz vorgestellt und in seiner Funktion als linearer Klassifikator erläutert. Anschließend wird die Verkettung von Perzeptronen zur Verwendung bei der Mehrklassen-Klassifikation erarbeitet. Im Anschluss daran wird ausgeführt, wie ein neuronales Netz lernt, für gegebene Daten eine korrekte Klassifikation durchzuführen.
In Abschnitt 2.2 wird die in dieser Arbeit verwendete Variante von neuronalen Netzen, sogenannte *Convolutional Neural Networks*, vorgestellt. Dabei wird der Fokus auf die namensgebende Faltungsoperation und die Repräsentation von Bildmerkmalen gelegt.

2.1 Neuronale Netze

2.1.1 Neuronen & Perzeptronen

Der Begriff *Neuronales Netz* hat seinen Ursprung im Versuch, eine mathematische Repräsentation von Informationsverarbeitung in biologischen Systemen zu finden. Die bereits in den 1940er Jahren begonnene und bis heute andauernde Entwicklung solcher Repräsentation bietet nur eine begrenzte biologische Plausibilität, welche zusätzliche Einschränkungen in der Anwendung zur Klassifikation mit sich bringen würde [Bishop, 2006, S.226]. Dennoch eignen sich biologische Neuronen als Inspiration für den Grundbaustein neuronaler Netze.
Um die Verarbeitung der Signale in einem Neuron mathematisch darstellen zu können, wird zunächst folgende Notation eingeführt: $\underline{\mathbf{x}}$ beschreibt als Vektor die Eingangssignale in einem Neuron. Die einzelnen Signale können als Skalare auch durch eine Indizierung entsprechend ihrer Herkunft notiert werden; so ist $\underline{x}_i$ das Eingangssignal, welches dem vorangegangenen i-ten Neuron entspringt. Die Eingangssignale werden jeweils mit einem Gewicht $\underline{w}_{ij}$ multipliziert, wobei der Index j das aktuell betrachtete Neuron beschreibt. Zusätzlich besitzt jedes Neuron einen Bias bzw. Schwellwert b. Aus diesen eingeführten Größen lässt sich die *Aktivierung* a_j des j-ten Neurons berechnen:

$$a_j = \sum_i^I \underline{w}_{ij} \cdot \underline{x}_i + b_j = \underline{\mathbf{w}}_j^T \cdot \underline{\mathbf{x}} + b_j \tag{2.1}$$

Es erfolgt also eine Aufsummierung aller mit den Gewichten multiplizierten Eingangssignalen sowie eine Addition des Bias [Bishop, 2006, S. 227]. Um die Darstellung zu vereinfachen, kann der Bias als Teil der Gewichte dargestellt werden. Dies ist vor allem deshalb sinnvoll, weil der daraus entstehende Vektor alle veränderlichen Größen eines Neurons enthält, die später gelernt werden. Damit die Vektormultiplikation gültig bleibt, wird der Vektor der Eingangssignale zusätzliche um eine 1 ergänzt. Damit ergibt sich:

$$\mathbf{w}_j = [\underline{\mathbf{w}}_j^T, b_j]^T \tag{2.2}$$

$$\mathbf{x} = [\underline{\mathbf{x}}^T, 1]^T \tag{2.3}$$

$$a_j = \mathbf{w}_j^T \cdot \mathbf{x} \tag{2.4}$$

Das tatsächliche Ausgangssignal o_j des Neurons ergibt sich dann durch das Anwenden einer ableitbaren, nicht-linearen Aktivierungsfunktion f auf die Aktivierung a_j eines Neurons [Bishop, 2006, S. 227]:

$$o_j = f(a_j) = f(\mathbf{w}_j^T \cdot \mathbf{x}) \tag{2.5}$$

Prinzipiell kann die Aktivierungsfunktion beliebig gewählt werden, bei neuronalen Netzen werden beispielsweise die logistische Sigmoidfunktion σ, die ReLU-Funktion (*recitified linear unit*) oder die Stufenfunktion τ genutzt:

$$\sigma(a) = \frac{1}{1 + e^{-a}} \tag{2.6}$$

$$ReLU(a) = max(0, a) \tag{2.7}$$

$$\tau(a) = \begin{cases} +1 \text{ wenn } a > 0 \\ -1 \text{ wenn } a < 0 \end{cases} \tag{2.8}$$

Die Stufenfunktion τ kann damit beispielsweise zur Klassifikation eines Zwei-Klassen-Problems genutzt werden, indem ein Ausgangssignal $o_j = \tau(\mathbf{w}_j^T \cdot \mathbf{x})$ entweder $+1$ oder -1 ergibt. So kann die Zugehörigkeit der Daten x zu einer von zwei Klassen bestimmt werden. Die durch die Vektormultiplikation (zwischen Gewichten und Eingangssignalen) beschriebene Entscheidungsgrenze für die Klassifikation stellt den Fall dar, dass die beide Klassen gleich wahrscheinlich sind. Diese Entscheidungsgrenze liegt für die Stufenfunktion bei $a_j = 0$. Diese Eigenschaft wird beim Perzeptron als einfachstes neuronales Netz zur Klassifikation genutzt. In Abbildung 2.1 wird dargestellt, wie ein Perzeptron dadurch in der Lage ist, eine lineare Klassifikation durchzuführen.

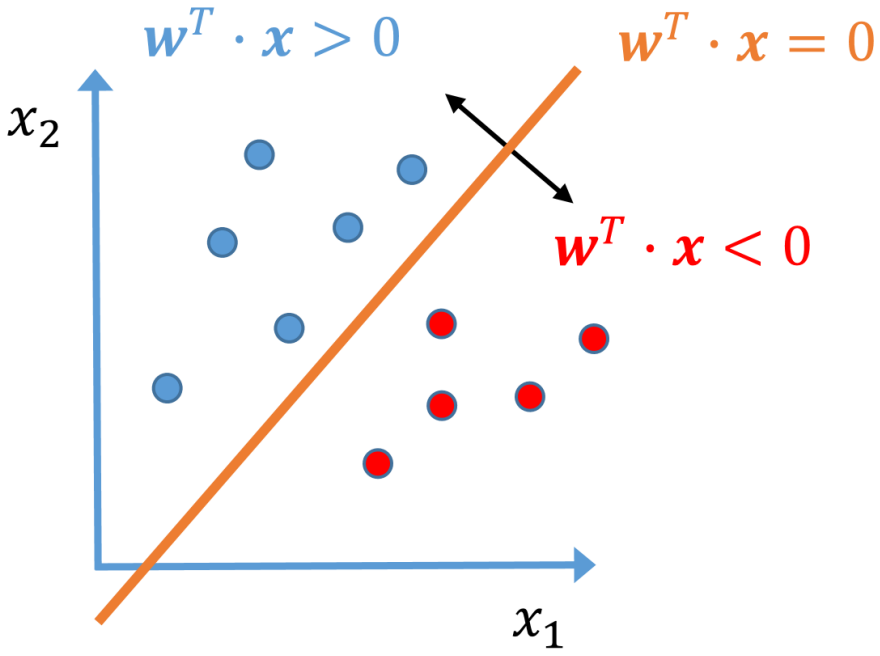

Abbildung 2.1: Lineare Klassifikation durch ein Perzeptron

Die Darstellung zeigt, wie ein Perzeptron für eine lineare Klassifikation verwendet werden kann. Eingangsdaten **x** links der Geraden erzeugen durch die Stufenfunktion einen Ausgangswert von +1, Daten rechts der Geraden entsprechend einen Ausgangswert von -1. Damit lassen sich die Eingangsdaten in zwei Klassen unterteilen.

Da Eingangsdaten oftmals nicht linear trennbar sind und häufig zwischen mehr als zwei Klassen unterschieden werden soll, würde ein einzelnes Perzeptron bei der Klassifikation nicht-linearer Daten versagen. Um dieses Problem zu beheben, können mehrere Neuronen zu einem *Multilayer-Perzeptron* verkettet werden, sodass der Ausgang eines Neurons den Eingang eines weiteren bildet. Dazu werden mehrere Neuronen in Schichten bzw. Layern angeordnet. Bei dieser sogenannten Feed-Forward-Architektur werden die D-dimensionalen Eingangsdaten ab dem *Input-Layer* durch eine oder mehrere Schichten, sogenannte *Hidden Layers*, propagiert. Jedes Layer kann dabei beliebig viele Neuronen beinhalten. Am Ende des Netzwerkes befindet sich das *Output-Layer*, in der Regel mit K Neuronen, wobei K die Anzahl der zu unterscheidenden Klassen ist. Somit kann der Ausgangswert der Neuronen im letzten Layer als Zugehörigkeit der Daten zu der jeweiligen Klasse interpretiert werden; je größer der Wert des k-ten Neurons, desto größer ist die vom Netz berechnete Wahrscheinlichkeit, dass die Ein-

gangsdaten zur k-ten Klasse gehören. In Ergänzung zur vorgestellten Notation für Neuronen wird das Ausgangssignal der letzten Neuronen, also im Output-Layer, mit $\underline{y_k}$ bezeichnet. Damit ergibt sich die Klassenzugehörigkeit zur k-ten Klasse, abhängig von allen Gewichten im Netz $\mathbf{w}$ und den Eingangsdaten $\mathbf{x}$, zu $\underline{y_k}(\mathbf{w}, \mathbf{x})$. In Abbildung 2.2 ist ein Multilayer-Perzeptron schematisch dargestellt. Die Architektur des Netzes, also die Anzahl und Größe der Hidden Layer sowie der Grad der Verknüpfung zwischen den Neuronen benachbarter Schichten, kann prinzipiell frei gewählt werden. Bei sogenannten *Fully Connected Layern* ist jedes Neuron mit jedem Neuron des nachfolgenden Layers verknüpft.

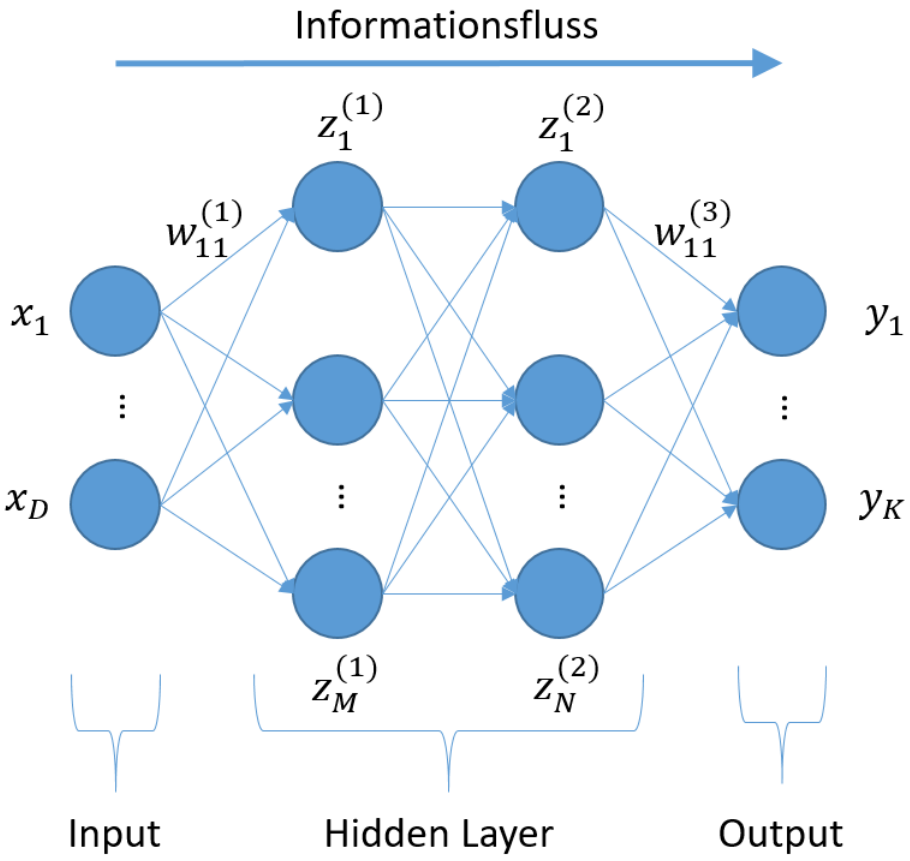

Abbildung 2.2: Multilayer-Perzeptron

Die Darstellung zeigt, wie die D-dimensionalen Eingangsdaten ab dem Input-Layer durch die beiden nachfolgenden Hidden Layer propagiert werden. Im Output-Layer befinden sich K Neuronen, je eines für jede mögliche Klasse.

Damit die Ausgangsgrößen aus dem Output-Layer als Zugehörigkeit zu bzw. Wahrscheinlichkeit für eine Klasse interpretiert werden können, wird die Softmax-Funktion auf die Ausgänge des Netzes angewendet [Bishop, 2006, S. 228]:

$$y_k = Softmax(\underline{y}_k(\mathbf{x}, \mathbf{w})) = \frac{e^{\underline{y}_k}}{\sum_{\kappa=1}^{K} e^{\underline{y}_\kappa}} \tag{2.9}$$

Das Ziel des neuronalen Netzes ist es nun, für einen gegebenen Input $\mathbf{x}$ dem Ausgangsneuron $\mathbf{y_k}$ die höchste Wahrscheinlichkeit zuzuweisen, zu dessen Klasse $\mathbf{x}$ gehört. Um diese Aufgabe zu erfüllen, werden die Gewichte des Netzes $\mathbf{w}$ auf geeignete Weise gelernt. Wie dieses Training abläuft, wird im folgenden Abschnitt erklärt.

2.1.2 Training

Für das Training eines Netzes sind N mehrdimensionale Trainingsbeispiele $\mathbf{x}_n$ gegeben. Jedes Trainingsbeispiel stellt D-dimensionale Eingangsdaten $\mathbf{x}_n = [x_{n,1}, ..., x_{n,D}]$ dar und besitzt außerdem einen Klassenzugehörigkeitsvektor $\mathbf{C}_n$:

$$\mathbf{C}_n = [C_n^1, ..., C_n^K]^T \tag{2.10}$$

$$C_n^k \in \{0,1\} \tag{2.11}$$

In dieser 1-in-K-Darstellung ist $C_n^k = 1$, wenn $\mathbf{x}_n$ zur Klasse C^k gehört. Für jedes der gegebenen Trainingsbeispiele $\mathbf{x}_n$ kann das Netz nun die Zugehörigkeit $y_{nk}(\mathbf{w}, \mathbf{x}_n) = Softmax(y_k(\mathbf{x}_n, \mathbf{w}))$ zur Klasse C^k berechnen.
Für das Training des neuronalen Netzes wird nun eine Verlustfunktion $E_n(\mathbf{w}, \mathbf{x}_n)$ eingeführt. Sie ist ein Maß dafür, wie sehr die prädizierte Klassenzugehörigkeit eines Datums von der Tatsächlichen abweicht. Eine Möglichkeit, dieses Maß zu berechnen, ist es, die Quadratsumme der Klassifikationsfehler zu bilden [Bishop, 2006, S. 233]:

$$E_n(\mathbf{w}, \mathbf{x}_n) = \frac{1}{2} \sum_{k=1}^{K} (y_{nk}(\mathbf{w}, \mathbf{x_n}) - C_n^k)^2 \tag{2.12}$$

Alternativ dazu kann die Kreuzentropie für die Verlustfunktion verwendet werden [Bishop, 2006, S. 235]. Diese ergibt sich zu:

$$E_n(\mathbf{w}, \mathbf{x}_n) = -\sum_{k=1}^{K} C_n^k \cdot ln(y_{nk}) \tag{2.13}$$

Nach [Simard et al., 2003] führt die Verwendung der Kreuzentropie anstelle der Quadratsumme der Fehler zu schnellerem Training und verbesserter Generalisierung. Das Ziel beim Training des neuronalen Netzes kann nun so interpretiert werden, dass die Verlustfunktion über alle Trainingsdaten minimal werden soll [Bishop, 2006, S. 235]:

$$E(\mathbf{w}) = \sum_{n=1}^{N} E_n(\mathbf{w}, \mathbf{x}_n) = -\sum_{n=1}^{N}\sum_{k=1}^{K} C_n^k \cdot ln(y_{nk}) \rightarrow min \tag{2.14}$$

Gesucht sind also Werte für die Gewichte $\mathbf{w}$, durch welche die Verlustfunktion $E(\mathbf{w})$ minimal wird. Die Anpassung der Gewichte in Bezug auf die Minimierung erfolgt durch ein iteratives Gradientenabstiegsverfahren. Dabei wird der Gradient der Verlustfunktion $\nabla E(\mathbf{w}^{(\tau)})$ für die aktuellen Gewichte $\mathbf{w}^{(\tau)}$ berechnet; der Gradient zeigt in die Richtung des steilsten Anstiegs der Verlustfunktion. Um diese zu minimieren, werden die Gewichte in Gegenrichtung zum Gradienten geändert, sodass sich neue Gewichte $\mathbf{w}^{(\tau+1)}$ ergeben [Bishop, 2006, S. 240]:

$$\mathbf{w}^{(\tau+1)} = \mathbf{w}^{(\tau)} + \Delta\mathbf{w}^{(\tau)} = \mathbf{w}^{(\tau)} - \eta\nabla E(\mathbf{w}^{(\tau)}) \tag{2.15}$$

Dabei ist η die sogenannte Lernrate, welche ein Steuerparameter dafür ist, wie sehr die Gewichte in jeder Trainingsiteration τ verändert werden. Abbildung 2.3 stellt einen Iterationsschritt des Gradientenabstiegsverfahrens dar. Ebenso sind dort ein lokales Minimum $\mathbf{w}_l$, ein Sattelpunkt $\mathbf{w}_s$ sowie das globale Minimum $\mathbf{w}_g$ der Verlustfunktion hervorgehoben. Während das Ziel des Trainings ist, die optimalen Gewichte $\mathbf{w}_g$ zu finden, an denen die Verlustfunktion so klein wie möglich ist, stellen lokale Minima und Sattelpunkte ein Problem dar. An diesen Punkten ist der Gradient gleich Null, es findet also keine weitere Veränderung der Gewichte statt, obwohl die optimale Lösung noch nicht gefunden wurde. Das Gradientenabstiegsverfahren konvergiert damit zu einer Lösung für die Gewichte $\mathbf{w}$, durch welche keine optimale Klassifikation möglich ist.
In der ersten Trainingsiteration $\tau = 1$ ist es notwendig, dass die Anfangswerte der

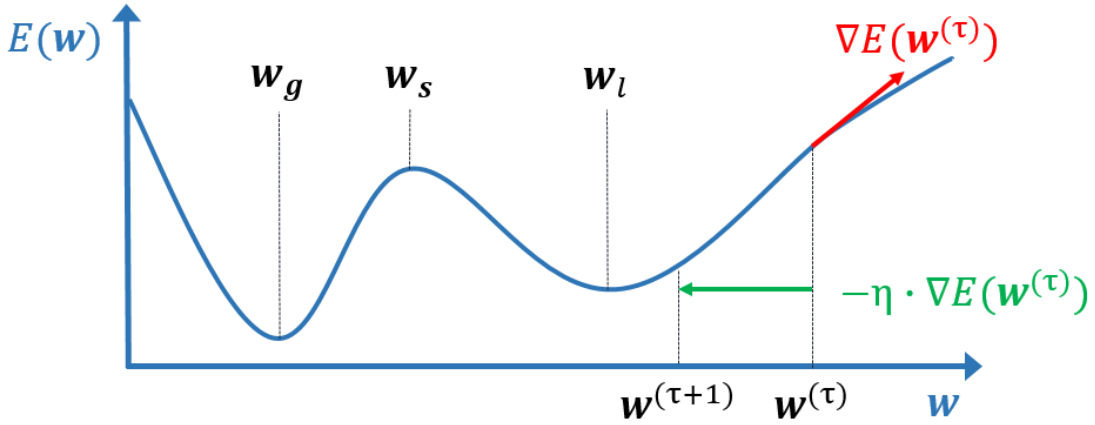

Abbildung 2.3: Gradientenabstiegsverfahren

Die Abbildung zeigt, wie durch eine Änderung der Gewichte (grün), abhängig vom Gradienten (rot), die Verlustfunktion verringert werden kann.

Gewichte $\mathbf{w}^{(0)}$ bereits gesetzt sind. Von der Wahl der initialen Werte hängt ab, ob das Training schnell zum Optimum führt und ob für alle Gewichte die optimalen Werte zur nahezu selben Trainingsiteration gefunden werden. Um diese Eigenschaften zu erfüllen, werden die Anfangswerte abhängig von der Anzahl der eingehenden Signale in einem Neuron gesetzt, sodass der Anfangswert eines Gewichts w_{ij} im Bereich $\frac{-1}{\sqrt{d_j}} < w_{ij} < \frac{+1}{\sqrt{d_j}}$ liegt. Dabei ist d_j die Anzahl der eingehenden Signale in das j-te Neuron [Duda et al., 2012].

Die oben beschriebene Variante des Gradientenabstiegsverfahrens ist eine Batch-Methode, da alle N verfügbaren Trainingsbeispiele zur Berechnung der Verlustfunktion $E(\mathbf{w})$ verwendet werden. Alternativ dazu kann nur immer ein einzelnes, zufällig gezogenes Beispiel $\mathbf{x}_n$ verwendet werden. Diese Variante, welche ein stochastisches Gradientenabstiegsverfahren darstellt, hat den Vorteil, dass stationären Punkten, also lokalen Minima oder Sattelpunkten, entkommen werden kann, da ein stationärer Punkt eines einzelnen Trainingsbeispiels nicht notwendig einem stationären Punkt aller Beispiele entspricht [Bishop, 2006, S. 241]. Ein Nachteil dieser Variante ist hingegen, dass die Anpassung der Gewichte in jeder Trainingsiteration nur anhand eines einzelnen Trainingsbeispiels erfolgt. So können sich die aus unterschiedlichen Beispielen

ergebenden Gradienten gegenseitig teilweise aufheben, was das Training verlangsamt. Als Variante zwischen Batch-Methode und stochastischer Methode gibt es außerdem die Mini-Batch-Methode, bei der M zufällige Beispiele $\mathbf{x}_m$ gezogen werden, anhand derer die Verlustfunktion berechnet und die Gewichte angepasst werden. Analog zur stochastischen Methode können hier die Gradienten verrauscht sein und so das Training verlangsamen, wenn die Größe des Mini-Batches M zu klein ist.
Eine Eigenschaft von Batch-Methoden in Zusammenhang mit der in Gleichung 2.14 vorgestellten Verlustfunktion ist es, dass häufig auftretende Klassen den Betrag der Verlustfunktion zu dominieren vermögen. Kommt in einem Batch beispielsweise die Klasse C^1 100-mal so häufig vor wie die Klasse C^2, so wird der Anteil der ersteren Klassen an der Verlustfunktion im Vergleich zur letzteren deutlich höher sein. Im Rahmen der Minimierungsaufgabe werden also tendenziell zuerst die Klassifikationsfehler 'behoben', die durch die häufiger auftretende Klasse C^1 verursacht werden. Der Netz lernt damit vorrangig, korrekte Prädiktionen bezüglich der Klasse C^1 zu berechnen. Korrekte Prädiktionen bezüglich der Klasse C^2 können so, wenn überhaupt, nur durch ein sehr langes Training erreicht werden.

Eine Problematik, die beim Training neuronaler Netze auftreten kann, ist die Überanpassung. Dabei werden die Gewichte während des Trainings so angepasst, dass die Verlustfunktion über alle Trainingsdaten möglichst klein wird; werden dem Netz dann Daten präsentiert, auf denen es bis dahin nicht gelernt hat, wird es diese Daten falsch klassifizieren, obwohl die Verlustfunktion auf den Trainingsdaten minimiert wurde. Das Netz hat dann die Trainingsdaten 'auswendig gelernt' und keine generalisierte Darstellung gefunden; es ist also an die Trainingsdaten überangepasst. Um diesem Problem entgegenzuwirken, werden Methoden der Regularisierung genutzt. Eine Möglichkeit bietet das *Weight Decay*, welches ein Netz dahingehend einschränkt, dass es die Trainingsdaten (durch Überanpassung der Gewichte) nicht auswendig lernt. Dazu wird die Verlustfunktion um die Quadratsumme der Gewichte des Netzes erweitert [Bishop, 2006, S. 257]:

$$E(\mathbf{w}) = \sum_{n=1}^{N} E_n(\mathbf{w}, \mathbf{x}_n) + \sum_{i,j} \mathbf{w}_{ij}^2 \cdot \lambda_w \tag{2.16}$$

Dabei ist λ_w ein Steuerparameter, mit welchem der Einfluss des *Weight Decay* kontrolliert werden kann. Eine weitere Möglichkeit zur Regularisierung stellt das *Early Stopping* dar [Bishop, 2006, S. 259]. Dazu wird neben dem Trainingsdatensatz ein Validierungsdatensatz benötigt, zu welchem für alle Daten die Klassenzugehörigkeit bekannt ist. In der Praxis werden alle verfügbaren Daten in einen Trainings- und einen Validierungssatz aufgeteilt. Während des Trainings kann damit überprüft werden, ob die Verlustfunktion auch für den von den Trainingsdaten unabhängigen Validierungssatz kleiner wird. Wenn die Verlustfunktion nur für die Daten des Trainingsdatensatzes und nicht mehr für die Daten des Validierungsdatensatzes besser werden, findet eine Überanpassung des Netzes statt. So können nach dem Training eines Netzes jene Gewichte als optimale Lösung angesehen werden, welche den geringsten Verlustwert auf dem Validierungsdatensatz erzeugen. Durch das Verwenden dieser Gewichte wird eine gute Generalisierung in Bezug auf die unabhängigen Validierungsdaten zu erzielt.

Die letzte hier vorgestellte Möglichkeit zur Regularisierung besteht in dem Nutzen von synthetischen Trainingsbeispielen [Bishop, 2006, S. 261]. Die Idee dahinter ist, dass sich die Klassenzugehörigkeit eines Datums bei bestimmten Transformationen nicht ändert. So zeigt zum Beispiel das Bild eines Autos auch dann noch ein Auto, wenn das Bild rotiert wird. Die Klassifikation des Bildes sollte also invariant gegenüber einer Rotation sein. Diese Eigenschaft kann für die Regularisierung genutzt werden, indem der Trainingsdatensatz um transformierte Kopien einzelner Trainingsbeispiele erweitert wird, wobei die Transformation die Klassenzugehörigkeit beibehält. Die Erweiterung der Trainingsdaten durch transformierte Kopien wird als Datenaugmentierung bezeichnet. Eine weitere Möglichkeit diese Transformationen zu berücksichtigen besteht darin, die Transformation in der Struktur des neuronalen Netzes einzubetten. Diese Art der Regularisierung findet vor allem bei *Convolutional Neural Networks* Anwendung, welche im folgenden Abschnitt vorgestellt werden.

2.2 Convolutional Neural Networks

Auf der Basis der Regularisierung wurden *Convolutional Neural Networks*(CNNs) durch [Le Cun et al., 1989] eingeführt, welche insbesondere bei der Klassifikation von Bildern

genutzt werden. Damit sind die Eingangsdaten von CNNs nicht bloß Vektoren (wie z.B. beim Multilayer-Perzeptron), sondern Matrizen, welche Bilder darstellen. Ein Eintrag in einer Eingangsmatrix ist damit der Grauwert eines Pixels des Eingangsbildes. Entsprechend können mehrere Eingangsmatrizen die Farbkanäle eines farbigen Eingangsbildes darstellen. Analog dazu sind die Neuronen der Hidden Layers eines CNNs als Ebenen angeordnet, sodass einzelne Neuronen als Pixel eines Bildes und die Ausgangssignale der Neuronen als Grauwerte der Pixel interpretiert werden können. Wie bei einem farbigen Eingangsbild mit z.B. drei Farbkanälen kann ein Hidden Layer aus mehreren Ebenen von Neuronen bestehen. CNNs nutzen die Eigenschaft von Bildern, dass benachbarte Grauwerte im Bild stärker korreliert sind als voneinander entfernte Grauwerte [Bishop, 2006, S. 267]: Die Eingangssignale eines Neurons stammen nur von Neuronen des vorherigen Layers, welche in der Nähe der gleichen Bild- bzw. Ebenenposition liegen. Damit erhalten einzelne Neuronen nur Signale aus einem lokalen rezeptiven Feld [Bishop, 2006, S. 268]. Eine weitere Eigenschaft von CNNs ist, dass die Neuronen einer Ebene in einem Layer die gleichen Gewichte für ihre Eingangssignale besitzen. Somit können die Gewichte dieser Neuronen als Faktoren einer Faltungsmatrix interpretiert werden, mit welcher das Eingangsbild gefaltet wird. Eine solche Faltung ist beispielhaft in Abbildung 2.4 dargestellt. Layer, in denen Neuronen Eingangssignale aufnehmen und daraus ein Ausgangssignal bzw. -bild berechnen, werden entsprechend als Convolutional Layer bezeichnet (Convolution $\hat{=}$ Faltung). Wie die Anzahl der Ebenen bzw. Faltungsmatrizen innerhalb eines solchen Layers kann auch die Größe des rezeptiven Feldes eines Neurons frei gewählt werden. Die Größe dieses rezeptiven Feldes entspricht dabei der Größe der Faltungsmatrix.

Ein weiteres Merkmals von CNNs sind die den Convolutional Layern folgenden *Pooling Layer*, in welchen die Auflösung des Ausgangsbildes verändert wird. Dazu wird zunächst ein Ausschnitt aus dem Ausgangsbild betrachtet, der zum Beispiel 2x2 Pixel groß ist. Aus diesen vier Pixeln wird ein neuer Wert berechnet und in einem nachfolgenden Merkmalsbild gespeichert. Wird zum Beispiel der größte der vier Pixelwerte gespeichert, handelt es sich um Max-Pooling, wird dagegen der Mittelwert gebildet, handelt es sich um Average-Pooling [Bishop, 2006, S. 269]. Danach wird der betrachtete Ausschnitt um eine wählbare Schrittweite weitergeschoben und es wird wieder ein Pooling-Wert berechnet So wird für das ganze Merkmals- oder Eingangsbild fortgefahren; bei der

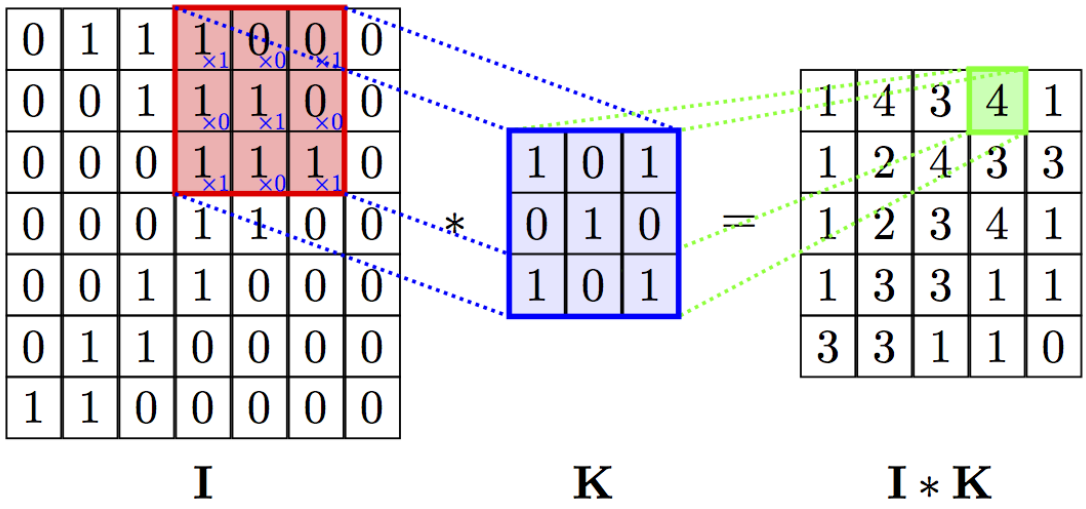

Abbildung 2.4: Faltung eines Bildes [PetarV, 2016]

Die Abbildung zeigt, wie das Eingangsbild I durch die Faltungsmatrix K gefaltet wird und somit ein Ausgangsbild I · K erzeugt. Die Koeffizienten der Faltungsmatrix K entsprechen den Gewichten, die das CNN während des Trainings lernt.

Verwendung eines 2x2-Poolings mit einer Schrittweite von zwei Pixeln wird die Größe des Bildes folglich um das vierfache verringert. In Abbildung 2.5 ist dieser Ablauf dargestellt.

Die durch die Abfolge von Convolutional und Pooling Layern gegebene Struktur hat die Eigenschaft, dass Merkmale im Eingangsbild, welche durch die Faltungsmatrizen z.B. in der oberen linken Ecke extrahiert werden, in den Ebenen der späteren Layers bzw. Ausgangsbildern auch in der linken oberen Ecke liegen. Die Topologie des Bildes bleibt also durch die Convolutional und Pooling Layer erhalten.

Da die Größe der Ausgangsbilder aufgrund der Pooling-Layer mit fortschreitender Tiefe des Netzes kleiner wird, bestehen die Bilder letztlich nur noch aus wenigen Pixeln, teilweise sogar nur aus einem. Doch obwohl die Pooling Layer die geometrische Größe des Bildes reduzieren, wird die Anzahl der Kanäle im Bild üblicherweise durch die Anzahl der Filtermatrizen in einem Layer gesteigert. Diese Ausgangsbilder können nun durch ihre Dimension als Merkmalsvektor des Eingangsbildes interpretiert werden, welcher die gelernten Eigenschaften des Bildes wiedergibt. Die Klassifikation des Bildes erfolgt dann, analog zum Multilayer-Perzeptron, durch einen oder mehrere Fully-Connected-

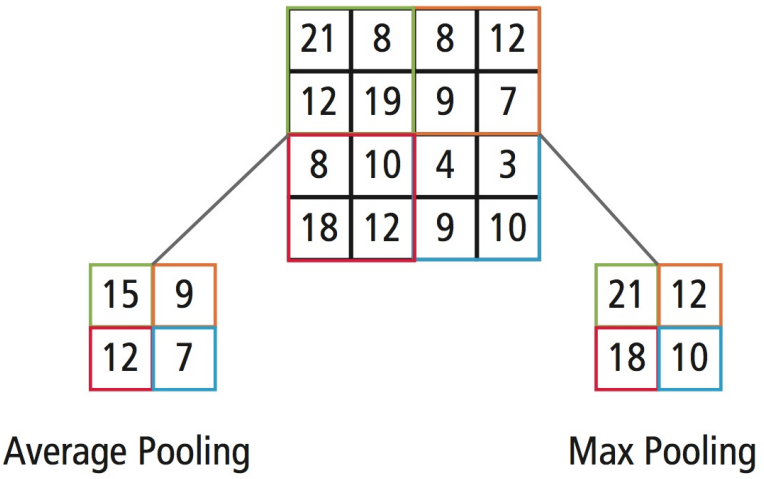

Abbildung 2.5: Pooling eines Bildes [Embedded Vision Alliance, 2019]

Die Abbildung zeigt, wie die Auflösung eines Bildes durch verschiedene Pooling-Varianten verringert werden kann. Die farbliche Einteilung gibt an, welche vier Pixelwerte zur Bildung des Durchschnitts oder des Maximums herangezogen werden.

Layer, endend mit einem Layer, welches so viele Neuronen besitzt, wie es Klassen zu unterscheiden gilt. Dieser Ablauf ist schematisch in Abbildung 2.6 dargestellt.

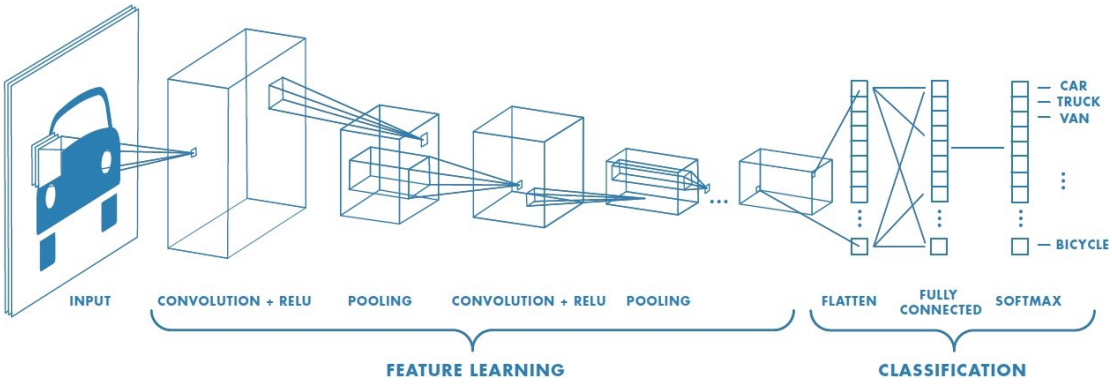

Abbildung 2.6: Abfolge der Layer in einem CNN [Prabhu, 2018]

Die Abbildung zeigt, wie aus einem Bild im ersten Abschnitt Merkmale extrahiert werden (Feature Learning). Im zweiten Abschnitt erfolgt die Klassifikation durch Fully-Connected-Layer, um letztendlich durch die Softmax-Aktivierungsfunktion die Klassenzugehörigkeit anzugeben.

KAPITEL 3

Methodik

In diesem Kapitel wird die in dieser Arbeit entwickelte Methodik ausführlich beschrieben. In Abschnitt 3.1 wird die Krater-Kandidaten-Auswahl beschrieben: Diese wird dazu verwendet, Positionen aus einem Kriegsluftbild zu extrahieren, an denen sich möglicherweise ein Bombenkrater befinden könnte. In Abschnitt 3.2 folgt die Erstellung der Datensätze für die neuronalen Netze, wozu die zuvor erarbeitete Krater-Kandidaten-Auswahl verwendet wird. Anschließend werden drei verschiedene Varianten von neuronalen Netzen vorgestellt, die sich in ihrer jeweiligen Architektur unterscheiden. Nachfolgend wird ausgeführt, wie diese Netze trainiert werden, wobei vor allem auf die Anpassung der Verlustfunktion eingegangen wird. Abschließend wird die Krater-Kandidaten-Auswahl mit der Klassifikation durch neuronale Netze kombiniert, um so die Detektion von Kratern in Bildern zu ermöglichen. In Abbildung 3.1 ist der Ablauf der entwickelten Methoden in einem Flussdiagramm dargestellt.

3.1 Krater-Kandidaten-Auswahl

Das Ziel bei der Krater-Kandidaten-Auswahl ist es, aus einem gegebenen Bild Koordinaten zu extrahieren, an denen sich Krater befinden könnten. Die extrahierten

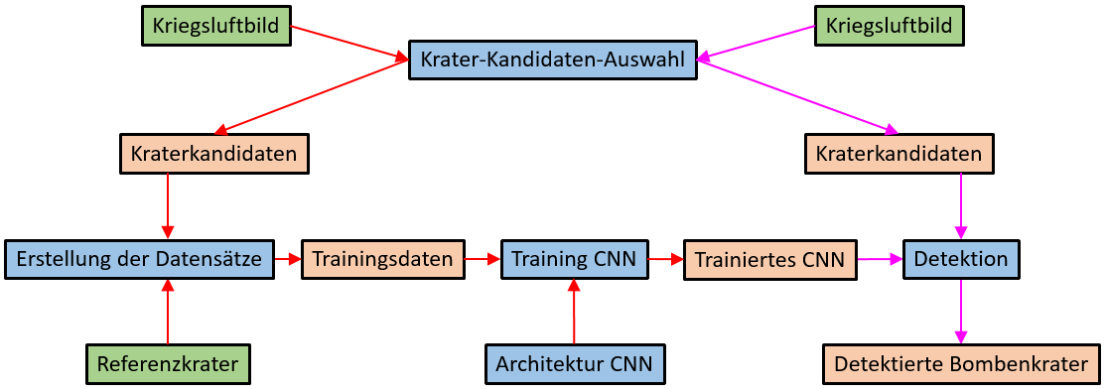

Abbildung 3.1: Ablauf der entwickelten Methoden

Die Abbildung stellt den Ablauf der Methoden als Flussdiagramm dar. Die Daten, die für diese Arbeit vom KBD Niedersachsen zur Verfügung gestellt wurden, sind grün markiert. Die blau markierten Elemente stellen die in diesem Kapitel entwickelten Methoden dar. Die während des Ablaufs erzeugten Daten sind orange markiert. Der durch die roten Pfeile beschriebene Pfad stellt den Ablauf der Methodik zum Trainieren der CNNs dar. Der durch die pinken Pfeile beschriebene Pfad stellt den Ablauf der Methodik zum Detektieren von Bombenkratern dar, nachdem die CNNs trainiert wurden. Am Ende des Diagramms stehen, wie es Ziel dieser Arbeit ist, in einem Kriegsluftbild detektierte Bombenkrater.

Koordinaten beschreiben mindestens die Position eines Kandidaten im Bild, wenn möglich auch dessen Größe. Diese Kandidaten werden zum einen dazu genutzt, um aus den gegebenen Daten Datensätze zum Trainieren und Testen der neuronalen Netze zu erstellen. Zum anderen sollen die trainierten neuronalen Netze später extrahierte Kraterkandidaten klassifizieren, sodass jeder Kandidat der Klasse *Krater* oder *Hintergrund* zugewiesen werden kann.

An die Krater-Kandidaten-Auswahl wird dabei die primäre Anforderung gestellt, dass alle Referenzkrater gefunden werden müssen. Diese Anforderung ist darum so wichtig, weil das neuronale Netz später nur dazu genutzt wird, die extrahierten Kandidaten zu klassifizieren. Das Netz kann also nicht dazu genutzt werden, weitere Kandidaten zu finden. Die Erfüllung dieser Anforderung kann dazu führen, dass Krater-Kandidaten an jeder möglichen Bildposition extrahiert werden. Da die Bombenkrater aber nur einen kleinen Teil eines Luftbildes ausmachen, würden damit deutlich mehr Kandidaten zur Klasse *Hintergrund* gehören. Da das Netz nach dem Training vermutlich nicht

jeden Kandidaten korrekt klassifizieren wird, führen mehr Kandidaten auch zu absolut mehr Fehlklassifikationen. Dieser Umstand wirkt sich negativ auf die in Abschnitt 4.4 erläuterte Richtigkeit aus. Da dies nicht erwünscht ist, wird an die Krater-Kandidaten-Auswahl die sekundäre Anforderung gestellt, dass so wenige Kandidaten wie möglich extrahiert werden.

Zu diesem Zweck wird für die Krater-Kandidaten-Auswahl der Blob-Detektor aus der OpenCV-Bibliothek [Bradski, 2000] verwendet. Diese Wahl basiert auf der Beobachtung, dass die Krater in den gegebenen Luftbildern als rundliche Blobs (Flecken) zu erkennen sind, welche in der Bildanalyse als Blobs bezeichnet werden. Zur Extraktion dieser Blobs geht der Detektor wie folgt vor: Zunächst wird das Grauwertbild, in welchem Blobs detektiert werden sollen, durch Schwellwertbildung in mehrere Binärbilder konvertiert. Der Schwellwert wird dabei, beginnend bei einer unteren Grenze von B_{minS}, für jedes Binärbild um eine Schrittweite B_{stepS} erhöht, bis eine obere Grenze von B_{maxS} erreicht wurde. In jedem dieser Binärbilder werden nun alle zusammenhängenden Pixel zu einer Gruppe zusammengefasst, welche als Binärblobs bezeichnet werden. Nachfolgend werden alle Binärblobs, die näher als $B_{minDist}$ zusammenliegen, zu einem Binärblob zusammengefasst. Abschließend wird für alle Binärblobs je der Mittelpunkt und der Radius bzw. die Größe berechnet. Die so erhaltenen Blobs können nun noch anhand verschiedener Kriterien gefiltert werden. Dabei wird später vor allem die Filterung anhand der minimalen (B_{minG}) und maximalen (B_{maxG}) Größe sowie die Filterung anhand der minimalen Rundlichkeit $B_{minRund}$, Konvexität $B_{minKonv}$ und Ausdehnung B_{minAus} eine wichtige Rolle spielen, um die Anzahl der detektierten Blobs zu beeinflussen.

3.2 Erstellung der Datensätze

Der Blob-Detektor wird dazu verwendet, um aus einem Bild eine Menge von Krater-Kandidaten zu extrahieren. Die Kandidaten werden genutzt, um aus ihnen Datensätze für das Trainieren, Validieren und Testen der neuronalen Netze zu erstellen. Dazu ist es nötig, für jeden extrahierten Kandidaten zu bestimmen, ob dieser tatsächlich einen Krater darstellt oder nicht. Angelehnt an Abschnitt 2.1 wird also jeder Kandidat mit einer entsprechender Klassenzugehörigkeit C_n^k versehen, wobei $C_n^{(1)} = 1$ die Zugehörigkeit

zur Klasse *Krater* und $C_n^{(2)} = 1$ die Zugehörigkeit zur Klasse *Hintergrund* beschreibt. Zu diesem Zweck werden die extrahierten Pixel-Koordinaten der Kandidaten mit den gegebenen Pixel-Koordinaten der Referenzkrater in den Bildern verglichen. Ist die Distanz δ zwischen einem Kandidaten und dem nächstgelegenen Krater kleiner als ein Schwellwert ϵ, so wird der Kandidat durch den Klassenzugehörigkeitsvektor $\mathbf{C}_n$ zur Klasse *Krater* gezählt, ansonsten zur Klasse *Hintergrund*:

$$\mathbf{C}_n = [C_n^{(1)}, C_n^{(2)}]^T = \begin{cases} [1,0]^T \text{ wenn } \delta < \epsilon \\ [0,1]^T \text{ wenn } \delta \geq \epsilon \end{cases} \tag{3.1}$$

Nun wird für jeden Kandidaten ein quadratischer Bildausschnitt um das Zentrum des Kandidaten gebildet. Dieser Ausschnitt hat eine Seitenlänge von 2ϵ, wodurch sichergestellt ist, dass die Darstellung im Ausschnitt der Klassenzugehörigkeit des jeweiligen Kandidaten entspricht. Die daraus resultierenden Ausschnitte stellen die mehrdimensionalen Eingangsdaten, also Eingangsbilder, $\mathbf{x}_n$ dar. In Abbildung 3.2 ist die Erstellung eines solchen Ausschnitts beispielhaft dargestellt.
Zusätzlich werden aus numerischen Gründen alle Daten $\mathbf{x}_n$, welche Bildausschnitte aus einem Grauwertbild mit Pixelwerten zwischen 0 und 255 darstellen, auf einen Wertebereich zwischen 0 und 1 transformiert.

3.3 Architektur der neuronalen Netze

Im folgenden Abschnitt werden drei Varianten von neuronalen Netzen vorgestellt, die für die Klassifikation von Kraterkandidaten genutzt werden sollen. Die erste Variante stellt ein selbst-erstelltes CNN mit wenigen Layern dar. Für die zweite Variante wird ein bereits vortrainiertes Netz zur Merkmalsextraktion genutzt. Die dritte Variante kombiniert beide Ansätze.
Was allen drei Varianten gemein ist, sind die jeweils drei letzten Layer: Diese bestehen zunächst aus zwei Fully-Connected (FC) Layern, wobei das erste 512 und das zweite 256 Neuronen besitzt, und abschließend aus einem Softmax-Layer, in denen die Klassenzugehörigkeit zu den beiden Klassen berechnet wird. In den beiden FC-Layern wird für alle Neuronen die ReLU-Aktivierungsfunktion verwendet, im Softmax-Layer wird dem Namen entsprechend die Softmax-Aktivierung verwendet. Die Unterschiede

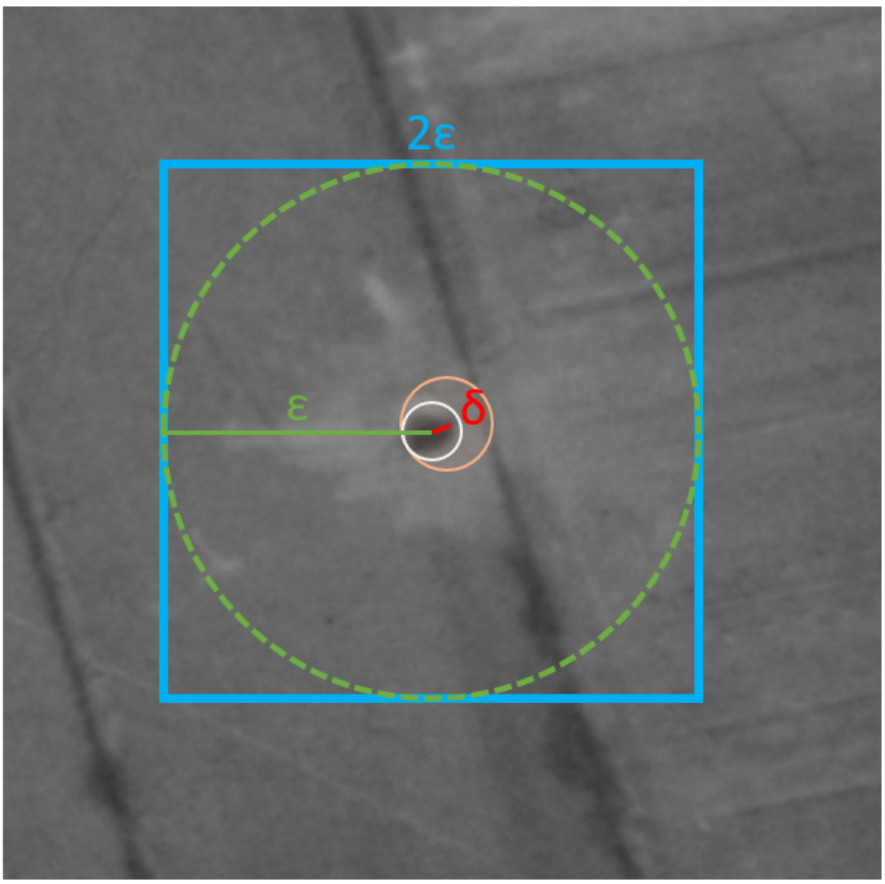

Abbildung 3.2: Bildausschnitt um einen Kraterkandidaten

Die Abbildung zeigt einen Teil eines Luftbildes, in dessen Mitte ein Bombenkrater zu sehen ist. Der orange Kreise stellt Position und Größe des Referenzkraters dar. Der weiße Kreis stellt Position und Größe des extrahierten Blobs, also des Krater-Kandidaten dar. Da der Abstand δ (rot) zwischen Kandidat und Referenz kleiner ist als der Grenzwert ϵ (grüner Kreis), wird dem Kandidaten die Klasse *Krater* zugewiesen. Das blaue Quadrat stellt den Bildausschnitt dar, der für diesen Kandidaten gebildet und extrahiert wird. Die Pixel des Ausschnitts stellen zudem die Eingangsdaten $\mathbf{x}_n$ des Netzes für diesem Krater dar.

zwischen den drei Varianten liegen somit vor allem in den vorderen Layern, also in der Merkmalsextraktion, während die Klassifikation durch die letzten drei Layer durch die gleiche Architektur in ihrer Komplexität vergleichbar bleibt.

Layer	Eingangsgröße	Dimension der Faltungsmatrizen	Pooling
1	160x160x1	(5x5) x 16	Ja
2	78x78x16	(7x7) x 32	Ja
3	36x36x32	(5x5) x 64	Ja
4	16x16x64	(5x5) x 128	Ja
5	6x6x128	(3x3) x 128	Nein
6	4x4x128	(4x4) x 256	Nein

Tabelle 3.1: Architektur zur Merkmalsextraktion in der ersten Netzvariante

Die Tabelle zeigt die Dimensionierung der Eingangsdaten sowie die Größe und Anzahl der Faltungsmatrien für jedes Layer. So besteht das Eingangsbild im ersten Layer aus 160x160 Pixeln mit einem Farbkanal, da das Eingangsbild ein Grauwertbild ist. Dieses wird durch 16 Faltungsmatrizen mit einer Größe von je 5x5 Pixeln gefaltet. Durch die Faltung wird die Größe des Eingangsbildes auf 156x156 Pixel reduziert. Nachfolgend wird Max-Pooling angewendet, wodurch die Größe des Bildes geviertelt wird. Daraus ergibt sich die Ausgangsgröße zu 78x78 Pixeln. Der Anzahl der Faltungsmatrizen entsprechend ergibt sich die Anzahl der Kanäle des Ausgangsbildes zu 16. Das Ausgangsbild des ersten Layers ist damit das Eingangsbild des zweiten Layers.

3.3.1 Variante 1: flaches CNN

Die erste Netzvariante besteht insgesamt aus nur 9 Layern, was im Vergleich zu modernen zur Bildklassifikation genutzten Netzen, wie z.B. *Inception-ResNet V2* [Szegedy et al., 2017] mit 467 Layern, relativ wenig ist. Die Idee dahinter ist, dass Krater im Bild eine einfache, kreisförmige Geometrie besitzen, weshalb keine komplexe Merkmalsextraktion nötig sein sollte, um die Kandidaten korrekt zu klassifizieren. Wie in Abschnitt 2.2 vorgestellt, verwendet auch diese Variante eine Abfolge von mehreren Convolutional und Pooling Layern. In allen Neuronen der Convolutional Layer wird die ReLU-Aktivierungsfunktion verwendet. In den Pooling Layern wird Max-Pooling mit einer Ausschnittsgröße von 2x2-Pixeln und einer Schrittweite von zwei Pixeln verwendet. Die genaue Architektur und Größe der einzelnen Faltungsmatrizen sind in Tabelle 3.1 angegeben. In Abbildung 3.3 ist das Netz schematisch dargestellt.

3.3.2 Variante 2: Merkmals-Extraktion

Diese Netzvariante macht Gebrauch vom *Transfer Learning*. Das bedeutet, dass ein Netz genutzt wird, welches bereits zur Klassifikation eines anderen Datensatzes trainiert

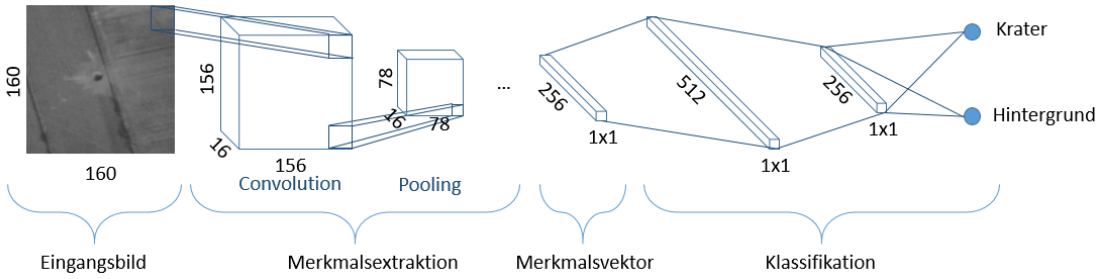

Abbildung 3.3: Schematische Darstellung des flachen CNN

Die Abbildung zeigt, wie im ersten Convolutional Layer aus dem Eingangsbild ein Merkmalsbild erstellt wird. Nachfolgend wird die Auflösung durch ein Max-Pooling-Layer verringert. Diese Abfolge wird wie beschrieben sechs Mal wiederholt, bis sich ein 256-dimensionaler Merkmalsvektor ergibt, durch welchen dann die Bestimmung der Klassenzugehörigkeit erfolgt.

wurde. Die Idee dabei ist, dass dieses Netz bereits gelernt hat, Merkmale zu extrahieren, die zur Unterscheidung der entsprechenden Klassen geeignet sind. Daher könnten diese Merkmale auch dafür geeignet sein, zwischen den hier vorliegenden Klassen *Krater* und *Hintergrund* zu unterscheiden.

Das verwendete, vortrainierte Netz ist das bereits erwähnte *Inception-ResNet V2* [Szegedy et al., 2017], welches für die *ILSVRC2015-Challenge* [Russakovsky et al., 2015] trainiert wurde. In dieser Challenge galt es, zwischen 1000 Klassen, wie z.B. verschiedenen Hunderassen oder Gebäudetypen, zu unterscheiden. Das Netz wird hier derart genutzt, dass aus einem Eingangsbild ein Merkmalsvektor extrahiert wird, der ursprünglich zur Klassifikation der 1000 Klassen verwendet wurde. Dieser 1536-dimensionale Merkmalsvektor wird dann durch die bereits vorgestellte Architektur von zwei FC-Layern und einem Softmax-Layer zur Klassifikation für das hier vorliegende Problem verwendet.

3.3.3 Variante 3: kombinierte Merkmals-Extraktion

Aufgrund der Größe des *Inception-ResNet V2* und der verwendeten Hardware ist es nicht möglich, die Gewichte dieses Netzes und damit die extrahierten Merkmale an das vorliegende Problem anzupassen. Daher stellt die dritte Netzvariante eine Kombination aus dem flachen CNN und dem Transfer Learning dar. Die Idee dabei ist, dass die

Gewichte des flachen CNNs durch das Training so angepasst werden, dass es Merkmale extrahiert, die eine für die Klassifikation nützliche Ergänzung zu den Merkmalen des *Inception-ResNet V2* darstellen. Hierzu werden die Merkmalsvektoren beider Netze kombiniert: Der kombinierte Merkmalsvektor M_K ergibt sich zu $M_K = [M_F^T, M_E^T]^T$, wobei M_F der Merkmalsvektor des flachen Netzes und M_E der Merkmalsvektor der Merkmals-Extraktions-Variante ist. Dieser kombinierte Merkmalsvektor wird durch die Endarchitektur von zwei FC-Layern und einem Softmax-Layer klassifiziert. Beim Training werden entsprechend nur die Gewichte des flachen CNNs angepasst. Die Architektur dieser kombinierten Variante ist in Abbildung 3.4 dargestellt.

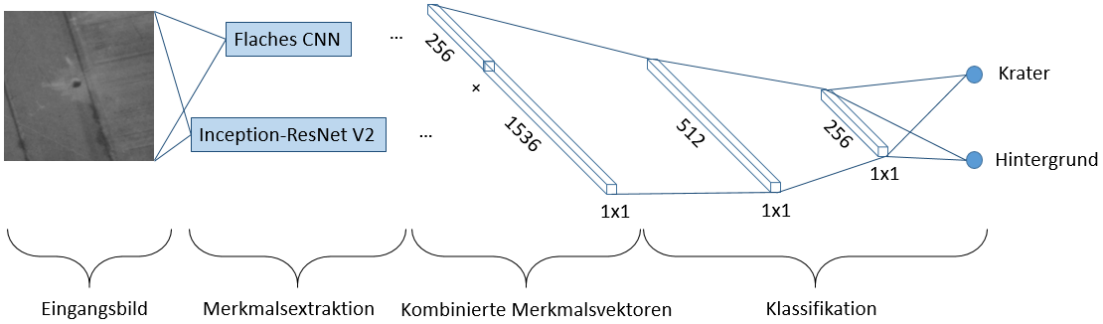

Abbildung 3.4: Schematische Darstellung des kombinierten Transfer Learning

Die Abbildung zeigt, wie die beiden vorgestellten Varianten kombiniert werden. Die aus dem Eingangsbild von beiden Netzvarianten extrahierten Merkmale werden kombiniert und zur Klassifikation genutzt.

3.4 Training der neuronalen Netze

Im folgenden Abschnitt wird das Vorgehen beim Training der neuronalen Netze beschrieben. Dieses wird für alle vorgestellten Varianten gleich gehandhabt.
Die Gewichte in den Netzen werden abhängig von der Anzahl der Eingänge eines Neurons initialisiert. Für die Neuronen in den letzten Layern, welche in allen Varianten gleich sind, wird dafür die in [He et al., 2015] vorgestellte Variance-Scaling-Methode verwendet, da diese sich vor allem für ReLU-Aktivierungsfunktionen in späteren Layern eignet. Für die restlichen Neuronen der Merkmalsextraktion, also für die Varianten 1 und 3, wird die

Xavier-Initialisierung verwendet, welche neben der Anzahl der Eingänge auch die Anzahl der ausgehenden Verbindungen eines Neurons betrachtet [Glorot & Bengio, 2015].
In jeder Trainingsiteration werden den Netzen 30 positive und 30 negative, zufällig ausgewählte Samples präsentiert. An dieser Stelle wird die in Unterabschnitt 2.1.2 vorgestellte *Data Augmentation* angewendet: Wenn die Samples aus dem Datensatz ausgewählt werden, wird zusätzlich eine zufällige Rotation um $0°, 90°, 180°$ oder $270°$ angebracht. Dadurch wird die Anzahl der im Datensatz verfügbaren Samples virtuell vervierfacht, was zu einer besseren Generalisierung beim Training führt.
Nachdem die Samples durch die Netze propagiert wurden, liegen für jedes Sample zwei Prädiktionen für die Klassenzugehörigkeiten vor; hierbei ist $y_n^{(1)}$ die prädizierte Klassenzugehörigkeit zur Klasse *Krater* $C^{(1)}$, $y_n^{(2)}$ entspricht der Zugehörigkeit zur Klasse *Hintergrund* $C^{(2)}$. Aus diesen Prädiktionen kann nun, wie in Unterabschnitt 2.1.2 beschrieben, die Verlustfunktion berechnet werden. Für die vorliegende Arbeit wird für die Verlustfunktion die Kreuzentropie aus Gleichung 2.14 verwendet. Zur Regularisierung wird die Verlustfunktion um einen Term zum *Weight Decay* ergänzt. Hierfür wird die in Unterabschnitt 2.1.2 vorgestellte Gleichung 2.16 zum *Weight Decay* um eine Normalisierung über die Summe aller P Gewichte w_p im Netz ergänzt. Dadurch ist der Anteil des *Weight Decay* an der Verlustfunktion für eine unterschiedliche Anzahl von Parametern in den Netzen vergleichbar. Dies ist vor allem wichtig, um den Einfluss des Steuerparameters λ_w für alle Netzvarianten vergleichbar zu machen, die sich durch den Unterschied in ihrer Architektur auch in der Anzahl der Gewichte unterscheiden. Abschließend wird die Verlustfunktion um den freien Parameter γ ergänzt, welcher den durch falsch-positive Klassifikationen erzeugten Fehler skaliert. Dieser Fehler wird in der Verlustfunktion durch $-C_n^{(2)} \cdot ln(y_n^{(2)})$ beschrieben: Liegt ein negatives Trainingsbeispiel vor ($C_n^{(2)} = 1$) und wird dieses vom Netz fälschlicherweise als positiv klassifiziert ($y_n^{(1)} = 1 \Rightarrow y_n^{(2)} = 0$), handelt es sich um eine falsch-positive Klassifikation. Für $\gamma > 1$ wird der Anteil dieser Fehler an der Verlustfunktion erhöht. Wie in Unterabschnitt 2.1.2 erklärt, lernt das Netz damit zuerst, solche Fehler zu vermeiden. Auf diese Weise wird ein größerer Anteil an negativen Trainingsbeispielen simuliert, ohne dass das Klassenverhältnis in einem Mini-Batch angepasst werden muss. Dies ist deswegen hilfreich, da aufgrund von Hardware-Beschränkungen nicht mit beliebig großen Mini-Batches und damit nicht mit beliebigen Klassenverhältnisse trainiert werden kann.

Durch die vorgestellten Ergänzungen ergibt sich die im Training zu minimierende Verlustfunktion $E(\mathbf{w})$ zu:

$$E(\mathbf{w}) = -\frac{1}{N}\sum_{n=1}^{N}\left\{C_n^{(1)} \cdot ln(y_n^{(1)}) + \gamma \cdot C_n^{(2)} \cdot ln(y_n^{(2)})\right\} + \frac{1}{P}\sum_{p=1}^{P} w_p^2 \cdot \lambda_w \qquad (3.2)$$

Für die Anpassung der Gewichte im Netz wird der *Adadelta*-Algorithmus [Zeiler, 2012] verwendet, welcher eine Erweiterung des normalen Gradientenabstiegsverfahrens darstellt. Bei der Berechnung der Änderung der Gewichte Δw berücksichtigt dieser Algorithmus einen exponentiell verfallenden Mittelwert der vorhergehenden quadrierten Gewichtsänderungen und Gradienten. Dadurch passt der Algorithmus die Größenordnung der Veränderung der Gewichte an die Topologie der Verlustfunktion an, was das Festlegen einer Lernrate erübrigt. Als freie Parameter dieses Algorithmus' bleiben damit die Verfallsrate ρ_{ada} für die Mittelwerte sowie eine Konstante ϵ_{ada} zur Konditionierung der Gradienten.
Als abschließende Maßnahme zur Regularisierung wird *Early Stopping* verwendet. Dazu wird in jeder 50. Trainingsiteration eine Validierung durchgeführt, wobei dem Netz analog zum Training je 30 positive und negative Samples präsentiert werden. Die Güte der Klassifikation wird anhand des F1-Maßes festgestellt, welches in Abschnitt 4.4 definiert wird. Bei jeder Validierung wird das erzielte F1-Maß mit dem zuletzt erreichten verglichen. Wenn sich die Güte der Klassifikation im Vergleich zur letzten Validierung verbessert hat, werden die aktuellen Gewichte des Netzes gespeichert. Damit wird sichergestellt, dass bei der späteren Verwendung der Netze zur Klassifikation im Detektor jene Gewichte verwendet werden, die auf einem unabhängigen Datensatz die beste Klassifikationsgüte erreicht haben. Das Training wird für eine feste Anzahl von Trainingsiterationen durchgeführt.

3.5 Detektion von Kratern

Die Detektion von Kratern in Luftbildern wird derart realisiert, dass die in Abschnitt 3.1 vorgestellte Methode genutzt wird, um Positionen aus einem Bild zu extrahieren, an denen sich möglicherweise ein Krater befindet. Diese Kraterkandidaten werden dann durch eine Netzvariante klassifiziert, um die Kandidaten den beiden Klassen

Krater oder *Hintergrund* zuzuweisen. Dabei wird für jeden Kandidaten die prädizierte Klassenzugehörigkeit $y_n^{(1)}$ gespeichert, welche der Wahrscheinlichkeit entspricht, dass der Kandidat einen Krater darstellt. Anschließend werden alle Kandidaten mit $y_n^{(1)} < \epsilon_{det}$ verworfen, wobei der Schwellwert ϵ_{det} einen freien Parameter darstellt.
Da die Kraterkandidaten als Bildausschnitte um eine Bildposition vorliegen, können die Kandidaten in ihrer Darstellung als Bounding-Box (BB) interpretiert werden. Aufgrund der in Abschnitt 3.2 festgelegten Ausschnittgröße ϵ haben alle BBs die gleiche Größe, gemessen an der Anzahl der Pixel im Bild. Da es durch die Krater-Kandidaten-Auswahl passieren kann, dass sich die Krater-BBs überschneiden, werden diese anhand ihrer Überlappung und ihrer jeweiligen Klassenzugehörigkeit $y_n^{(1)}$ gefiltert. Dazu wird zwischen allen Bounding-Boxes der Grad der Überlappung durch die *Intersection over Union* (IoU) bestimmt. Die IoU zwischen zwei Bounding-Boxes $BB_1 \& BB_2$ ergibt sich aus dem Anteil der Schnittmenge $I = BB_1 \cap BB_2$ an der Vereinigung $U = BB_1 \cup BB_2$:

$$IoU(BB_1, BB_2) = \frac{I}{U} = \frac{BB_1 \cap BB_2}{BB_1 \cup BB_2} \tag{3.3}$$

Liegt dieses Überschneidungsmaß über einem wählbaren Schwellwert ϵ_{IoU}, wird diejenige der beiden BBs verworfen, deren Klassenzugehörigkeit $y_n^{(1)}$ geringer ist. Alle Bounding-Boxes, die nach diesen Filterschritten noch vorhanden sind, gelten als im Bild detektierte Krater. In Abbildung 3.5 ist der Ablauf dieser Filterschritte dargestellt.

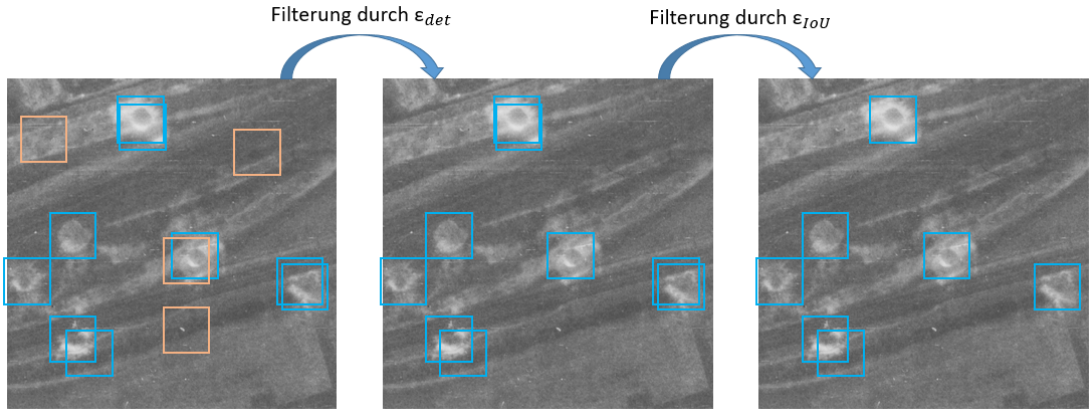

Abbildung 3.5: Detektion durch Filterung der Kandidaten

Die Abbildung zeigt, wie die klassifizierten Kraterkandidaten gefiltert werden. Im ersten Schritt werden Kandidaten anhand der prädizierten Klassenzugehörigkeit $y_n^{(1)}$ gefiltert. Für die blauen Kandidaten liegt diese über dem Schwellwert ϵ_{det}, für orange Kandidaten liegt sie darunter. Nachfolgend werden die Kandidaten anhand ihrer Überlappung und der verglichenen Klassenzugehörigkeit gefiltert. Ist die Überlappung zwischen zwei Kandidaten größer als ϵ_{IoU}, wird der Kandidat mit der kleineren Klassenzugehörigkeit $y_n^{(1)}$ verworfen. Ist die Überlappung kleiner als ϵ_{IoU}, bleiben beide Detektionen erhalten, wie es unten links im letzten Bild zu sehen ist.

KAPITEL 4

Evaluierung der entwickelten Methoden

In diesem Kapitel wird untersucht, inwiefern die entwickelten Methoden zur Detektion von Bombenkratern in historischen Luftbildern geeignet sind. Dazu wird zunächst in Abschnitt 4.1 auf die vom Kampfmittelbeseitigungsdienst Niedersachsen zur Verfügung gestellten Daten eingegangen. Hierbei steht die Qualität und die Art der Daten im Vordergrund. Anschließend beginnt mit Abschnitt 4.2 die experimentelle Untersuchung des für die Krater-Kandidaten-Auswahl gewählten Verfahrens. Hierfür werden Kriterien festgelegt, anhand derer diese Methodik bewertet wird. Darauf folgt mit Abschnitt 4.3 die Untersuchung, wie gut die zuvor entwickelte Krater-Kandidaten-Auswahl zum Erstellen der Datensätze genutzt werden kann. In Abschnitt 4.4 werden die verschiedenen Netzvarianten auf Grundlage von geeigneten Qualitätsmaßen bewertet und untereinander verglichen. Abschließend wird in Abschnitt 4.5 die Detektion von Bombenkratern (als Kombination aus Krater-Kandidaten-Auswahl und Klassifikation) anhand der zuvor eingeführten Qualitätsmaße untersucht und bewertet.

4.1 Analyse und Vorverarbeitung der Rohdaten

Für diese Arbeit wurden vom KBD Niedersachsen historische Luftbilder zur Verfügung gestellt. Für jedes Bild liegt eine Liste aller im Bild befindlicher Bombenkrater als Referenz vor. Zu jedem Referenzkrater ist sowohl die Position in Pixelkoordinaten als auch der Radius, angegeben in Pixeln, bekannt. Insgesamt stehen 22 Bilder zur Verfügung, die zwischen dem 09.09.1944 und dem 10.10.1945 von den Alliierten aufgenommen und später mit einer Auflösung von 1200dpi gescannt wurden. Die Bilder haben eine radiometrische Auflösung von 8-Bit und verschiedene Maßstäbe zwischen 1:6.000 und 1:27.000; dies entspricht einer Bodenpixelgröße (Ground Sampling Distance, GSD) zwischen 13 cm und 57 cm. In den Bildern befinden sich zwischen 0 und 988 Krater. Die untersuchten Bilder werden als repräsentativ angenommen, da unterschiedliche Belichtungsverhältnisse auftreten und der Inhalt der Bilder variiert. Die Bilder stellen nur ländliche Gebiete dar, da Bombenkrater in dicht bebauten städtischen Gebieten von Schutt verdeckt werden und somit nicht im Bild zu erkennen wären. In Abbildung 4.1 ist ein Beispiel eines genutzten Luftbildes mit eingezeichneten Referenzkratern abgebildet. Das dargestellte Beispiel macht zwei Problematiken bezüglich der Referenzkrater deutlich: Die Größe bzw. der Radius der Referenzkrater orientiert sich anscheinend nicht an der Form der Krater selbst, sondern an der Form der Bombenkrater zusammen mit deren resultierenden Bodenauswürfen. Diese Auswürfe sind jedoch für Krater gleicher Größe teilweise unterschiedlich groß. Damit kommt es vor, dass Krater mit augenscheinlich gleichem Radius jeweils durch Referenzkrater mit unterschiedlichem Radius dargestellt werden. Ebenso kommt es vor, dass der Mittelpunkt eines Referenzkraters nicht im augenscheinlichen Mittelpunkt eines Kraters liegt. Diese beiden Umstände stellen ein Problem dar, da die Referenzkrater sich für gleichartige Krater unterscheiden. Diese Problematik wird in Abschnitt 4.2 wieder aufgegriffen. Ein weiteres Problem betrifft die Qualität der Bilder: So erscheinen einige Bereiche in den Bildern zu hell bzw. überbelichtet, andere hingegen zu dunkel bzw. unterbelichtet. Damit wird in solchen Bereichen der Bilder nicht die volle radiometrische Auflösung ausgenutzt. Um diesem Problem entgegenzuwirken, wird eine lokale Histogramm-Einebnung verwendet, wie sie in [Bradski, 2000] durch die *Contrast Limited Adaptive Histogram Equalization* implementiert ist. Durch die Anwendung wirken die Bilder kontrastreicher,

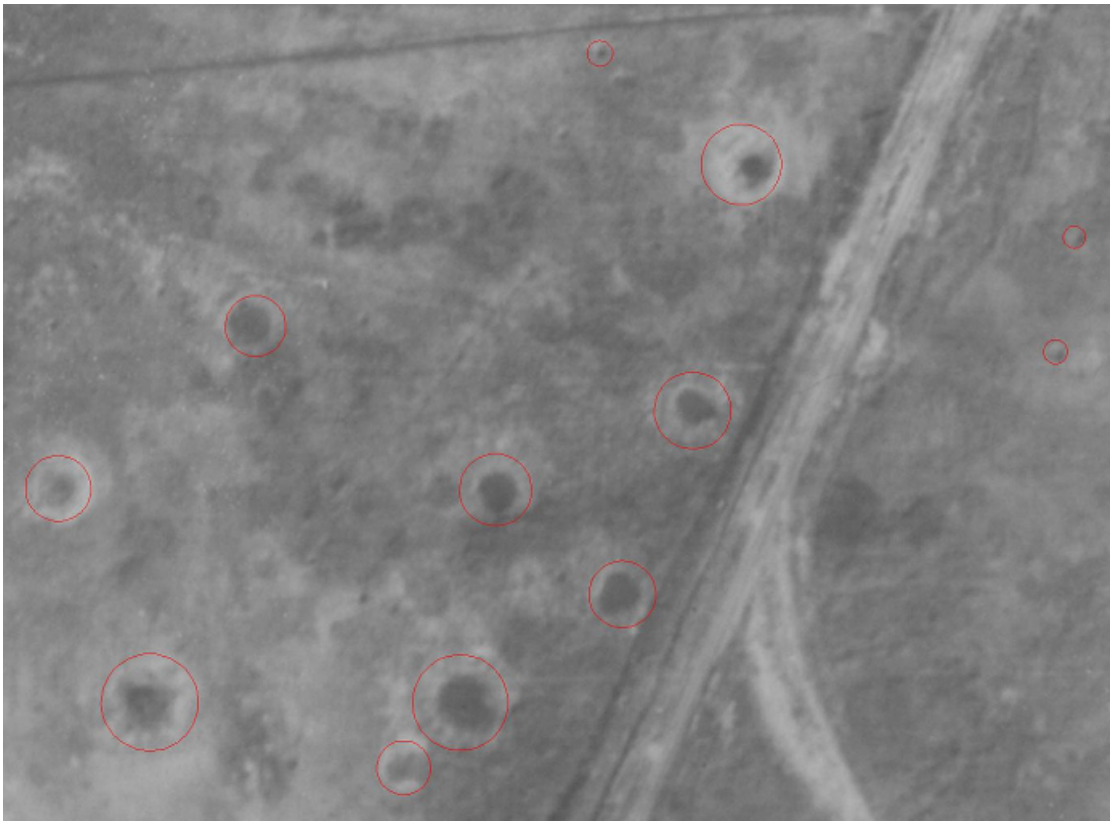

Abbildung 4.1: Luftbild mit eingezeichneten Krater-Referenzen

Das Bild zeigt einen Ausschnitt eines Luftbildes. Die roten Kreise stellen die gegebenen Referenzkrater dar.

wodurch Einschlagskrater augenscheinlich besser zu erkennen sind. Die Annahme ist, dass dies die Detektion der Bombenkrater positiv beeinflusst. Die Auswirkung der Histogramm-Einebnung ist in Abbildung 4.2 exemplarisch für ein Luftbild dargestellt. Für die nachfolgenden Analysen der gegebenen Daten werden diese zunächst in zwei getrennte Mengen aufgeteilt. Die eine Menge besteht aus 18 Bildern und den dazugehörigen Referenzkratern. Diese Menge wird verwendet, um den Einfluss der Parameter auf den methodischen Ablauf bis zur Detektion zu untersuchen; diese Daten werden bis einschließlich Abschnitt 4.4 genutzt. Die zweite Menge, bestehend aus vier Bildern, stellt einen unabhängigen Datensatz zum Testen der Kraterdetektion dar. In diesem kommen entsprechend keine Daten vor, die für das Training der CNNs verwendet wurden. Die vier gewählten Bilder unterscheiden sich im Grad ihrer Belastung: Im

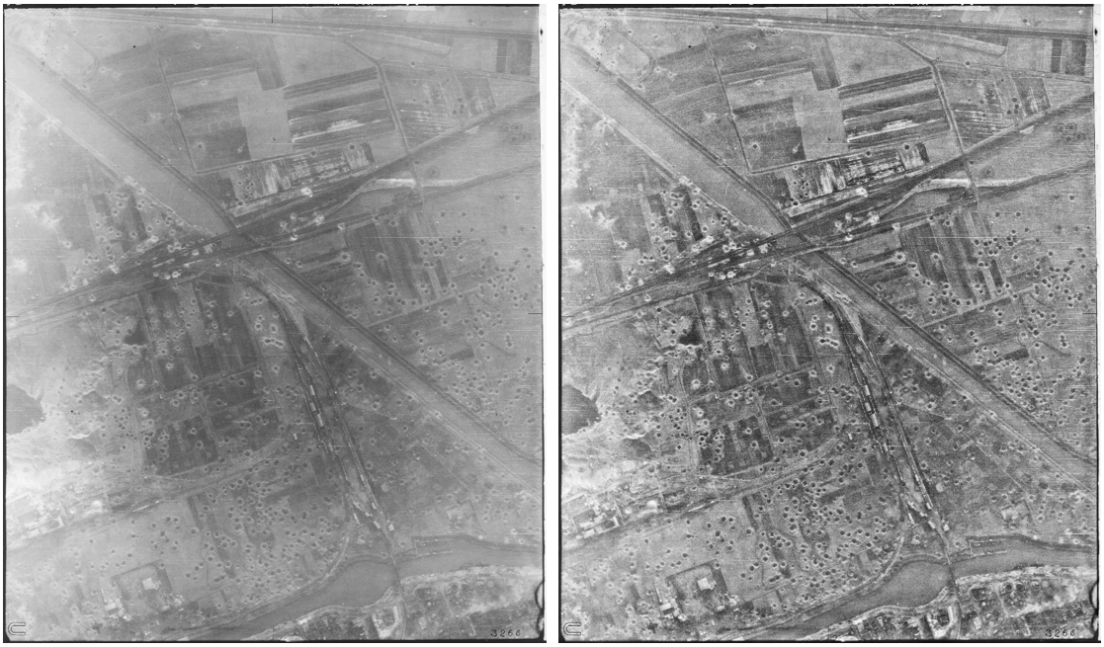

Abbildung 4.2: Auswirkung der lokalen Histogramm-Einebnung

Die Darstellung zeigt links ein Luftbild vor und rechts nach der lokalen Histogramm-Einebnung. Das rechte Bild wirkt kontrastreicher, einzelne Krater sind besser zu erkennen.

ersten Bild sind 801 Krater dargestellt, im zweiten und dritten je 18 und 16, im letzten Bild nur vier. Dadurch kann das Verfahren auch bezüglich der Belastung der Bilder untersucht werden. Mit dem Datensatz soll in Abschnitt 4.5 überprüft werden, ob die implementierten Methoden und deren Parameter so gewählt wurden, dass auch in bisher unbekannten Bildern Krater erfolgreich detektiert werden können.
Da die verwendeten Bilder jeweils unterschiedliche Maßstäbe aufweisen, ergibt sich folgender Umstand: Krater, die in der Realität gleich groß sind, erscheinen in den Bildern als unterschiedlich groß, da sich die Radien, gemessen in Pixeln, unterscheiden. Zudem unterscheiden sich Krater auch in der Realität in ihrer Größe, da verschiedene Bomben verschieden große Krater verursachen; dies trägt weiter zu den Größenunterschieden in Pixeln bei. Vor allem für die Krater-Kandidaten-Auswahl in Abschnitt 4.2, aber auch für die Erstellung der Datensätze in Abschnitt 4.3 ist es wichtig, die Verteilung

der Kratergrößen in den Bildern zu kennen: Diese Verteilung ist in Abbildung 4.3 als Histogramm dargestellt.

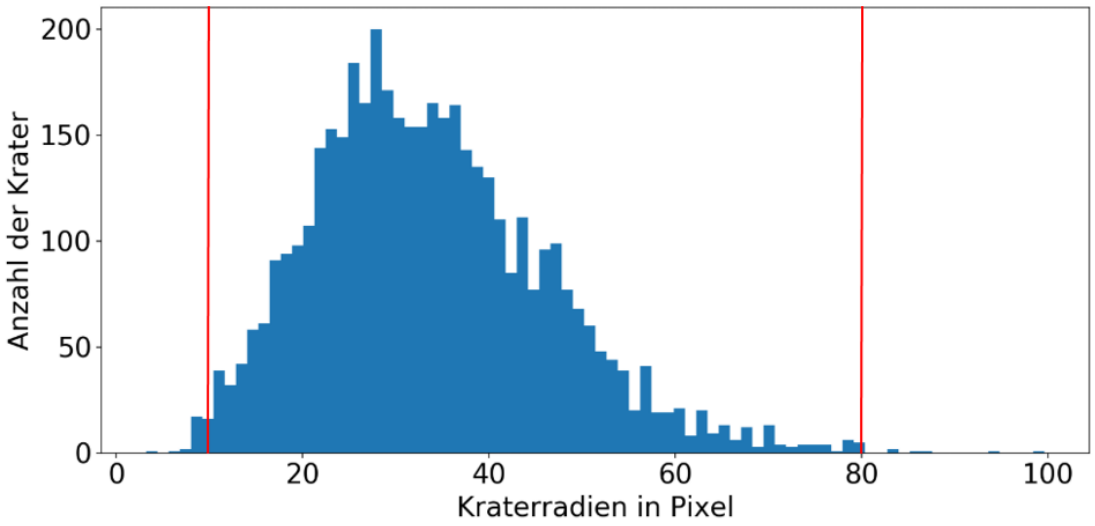

Abbildung 4.3: Histogramm der Kraterradien in Pixel

Die Grafik zeigt die Verteilung der Kraterradien. Der Bereich zwischen den roten Linien macht 99 % aller Krater aus.

4.2 Krater-Kandidaten-Auswahl

In diesem Abschnitt wird der Blob-Detektor für die Krater-Kandidaten-Auswahl untersucht. Wie zuvor erwähnt, wird der Blob-Detektor zum einen für die Erstellung der Datensätze für die CNNs genutzt. Zum anderen wird er später in Verknüpfung mit der Klassifikation durch CNNs zur Detektion von Bombenkratern verwendet. Das primäre Ziel für die Krater-Kandidaten-Auswahl ist es, möglichst alle Referenzkrater ($> 99\%$) in den Bildern zu finden. Die Anzahl aller Krater-Kandidaten gering zu halten ist dabei zweitrangig.

Für die Untersuchung der Krater-Kandidaten-Auswahl wird zunächst das Bewertungskriterium eingeführt, anhand dessen der Blob-Detektor beurteilt werden kann. In Anlehnung an Abschnitt 3.2 gilt eine Referenz dann als gefunden, wenn der Abstand zu einem detektierten Blob geringer ist als ϵ_{blob}. Motiviert aus der Verteilung der Krater-

größen (s. Abbildung 4.3) und der Erstellung des Trainingsbeispiele (s. Abbildung 3.2) wird $\epsilon_{blob} = 80$ px gesetzt. Bei der Evaluierung der Krater-Kandidaten-Auswahl ist zu beachten, dass ein Kandidat nur zu einer Referenz gehören kann; haben mehrere Referenzen einen Abstand von weniger als $\epsilon_{blob} = 80$ px zu einem Kandidaten, so gilt nur die nächstgelegene Referenz als zu einem Kandidaten gehörig. Die restlichen Referenzen müssen dann das Abstandskriterium auch zu anderen Kandidaten einhalten, um als gefunden zu gelten.

Im Folgenden wird die für die Krater-Kandidaten-Auswahl gewählte Methode des Blob-Detektors untersucht. Dazu werden die in Abschnitt 3.1 vorgestellten Parameter variiert und ihr Einfluss auf die Anzahl der gefundenen Referenzen sowie auf die Gesamtzahl aller Kandidaten untersucht.
Für die Experimente werden einige Parameter des Blob-Detektors nicht variiert, sondern vorher festgelegt. Die Bombenkrater haben in den Luftbildern ein eher dunkles Erscheinungsbild, welches sich durch die lokale Histogramm-Einebnung durch Grauwerte nahe 0 auszeichnet. Deshalb wird die untere Grauwertgrenze der Blobs zu $B_{minS} = 0$ festgelegt. Die in Abschnitt 4.3 ausgeführte Analyse der Kratergrößen ergab, dass die kleinsten Krater einen Radius von 10 Pixeln haben. Da es in dieser Größenordnung problematisch ist, sich überlappende Krater zu detektieren, wird der Mindestabstand zwischen den Blobs auf $B_{minDist} = 10$ festgesetzt. Zusätzlich lassen sich die Blobs durch ihre Größe filtern; dies erfolgt durch das Festlegen einer Anzahl von Pixeln, die zu einem Blob gehören. Aufgrund der rundlichen Erscheinung der Bombenkrater kann die Anzahl der Pixel im Blob durch eine Kreisfläche approximiert werden. So wird die maximale Größe der Blobs auf $B_{maxG} = 80^2 \cdot \pi \approx 20106$ festgelegt, ebenfalls motiviert durch die Analyse der Kratergrößen.
In Tabelle 4.1 sind die Ergebnisse der ersten Versuchsreihe zum Blob-Detektor aufgeführt. Hier wird zuerst versucht, die Blob-Detektion durch das Nutzen der Formfilter für minimale Rundlichkeit ($B_{minRund}$), minimale Konvexität ($B_{minKonv}$) und minimale Ausdehnung (B_{minAus}) einzuschränken. Der gewählte Wert für den maximalen Grauwert $B_{maxS} = 255$ ergibt sich daraus, dass in den Bildern teilweise auch hell erscheinende Krater vorkommen. Die Wahl für die minimale Blob-Größe $B_{minG} = 9$, hier wie oben als Radius angegeben, ergibt sich aus der Überlegung,

dass die Krater einen minimalen Radius von 10 Pixeln haben; da aufgrund des Grauwertverlaufs innerhalb des Kraters aber vermutlich nicht der gesamte Krater als Blob detektiert wird, resultiert daraus ein niedrigerer minimaler Radius. Die Ergebnisse zeigen, dass selbst ohne die Nutzung der Formfilter (Versuch 5) nicht über 99 % der Kraterreferenzen gefunden werden können.

Versuch	**1**	**2**	**3**	**4**	**5**
$\mathbf{B_{maxS}}$	255	255	255	255	255
$\mathbf{B_{stepS}}$	2	2	2	2	2
$\mathbf{B_{minG}}$	9	9	9	9	9
$\mathbf{B_{minRund}}$	0,7	0,5	0,3	0,1	0
$\mathbf{B_{minKonv}}$	0,7	0,5	0,3	0,1	0
$\mathbf{B_{minAus}}$	0,7	0,5	0,3	0,1	0
Vollständigkeit [%]	12,3	45,6	79,8	97,4	98,6
Anzahl Blobs	1299	9747	52836	251532	349222

Tabelle 4.1: Erste Versuchsreihe zum Blob-Detektor

Aus dieser Erkenntnis heraus wird in der nächsten Versuchsreihe in Tabelle 4.2 zunächst die minimale Blob-Größe weiter herabgesetzt. Die Vermutung dabei ist, dass die Grauwertverläufe innerhalb der Krater dazu führen, dass nur ein kleiner Teil im Innern der Krater als Blob detektiert wird; dieser Teil hat also einen kleineren Radius. Anschließend wird der Einfluss der einzelnen Formfilter für Rundlichkeit, Konvexität und Ausdehnung untersucht. Die Ergebnisse zeigen zunächst, dass das Herabsetzen der minimalen Größe auf $B_{minG} = 7$ dazu führt, dass die selbst gesetzte Anforderung hinsichtlich der Vollständigkeit erfüllt wird. Die Ergebnisse für den Einfluss der Formfilter zeigen, dass das Filtern der Rundlichkeit und Ausdehnung die Anzahl der Blobs zwar stark reduziert. Allerdings wird dann die Anforderung an die Vollständigkeit nicht weiter eingehalten. Die Konvexität wird also als einziger Formfilter genutzt, da diese die Vollständigkeit beibehält und die Anzahl der Blobs, wenn auch nur minimal, reduziert.

Die letzte Versuchsreihe in Tabelle 4.3 untersucht weitere Variationen für den maximalen Grauwert B_{maxS}, die Schrittweite B_{stepS} und die minimale Größe B_{minG}. Der Versuch 13 erreicht anhand der festgelegten Kriterien die besten Ergebnisse: Zum einen wird die Anforderung an die Vollständigkeit erfüllt. Zum anderen konnte durch diesen Versuch

Versuch	**6**	**7**	**8**	**9**
$\mathbf{B_{maxS}}$	255	255	255	255
$\mathbf{B_{stepS}}$	2	2	2	2
$\mathbf{B_{minG}}$	7	7	7	7
$\mathbf{B_{minRund}}$	0	0	0	0,3
$\mathbf{B_{minKonv}}$	0	0	0,3	0
$\mathbf{B_{minAus}}$	0	0,3	0	0
Vollständigkeit [%]	99,6	97,2	99,5	93.7
Anzahl Blobs	576779	281621	574730	225921

Tabelle 4.2: Zweite Versuchsreihe zum Blob-Detektor

Versuch	**10**	**11**	**12**	**13**
$\mathbf{B_{maxS}}$	255	220	220	220
$\mathbf{B_{stepS}}$	3	3	4	4
$\mathbf{B_{minG}}$	7	7	7	6
$\mathbf{B_{minRund}}$	0	0	0	0
$\mathbf{B_{minKonv}}$	0,3	0,3	0,3	0,3
$\mathbf{B_{minAus}}$	0	0	0	0
Vollständigkeit [%]	99,2	99,2	98,6	99,4
Anzahl Blobs	417046	416743	320927	430064

Tabelle 4.3: Dritte Versuchsreihe zum Blob-Detektor

die Anzahl aller Blobs im Vergleich zu Versuchen mit ähnlicher Vollständigkeit reduziert werden.

4.3 Datensätze für die CNNs

In diesem Abschnitt wird gezeigt, wie die oben vorgestellte Methode zur Krater-Kandidaten-Auswahl genutzt wird, um daraus Datensätze zum Trainieren, Validieren und Testen der neuronalen Netze zu erstellen. Zunächst werden alle der ca. 430.000 Kandidaten anhand des in Abschnitt 3.2 vorgestellten Kriteriums mit den gegeben Referenzen verglichen, sodass für jeden Kandidaten eine Klassenzugehörigkeit festgelegt wird. Daraus ergibt sich, dass insgesamt 18.816 positive Trainingsbeispiele und 411.248 negative Trainingsbeispiele für die Datensätze zur Verfügung stehen. In Abbildung 4.4 sind einige positive und negative Trainingsbeispiele dargestellt.
Die so erhaltenen, gelabelten Daten werden nun zufällig auf drei verschiedene Da-

tensätze aufgeteilt, genauer auf einen Trainings-, einen Validierungs- und einen Testdatensatz. Der Trainingsdatensatz wird für das Training der Netze verwendet. Der Validierungsdatensatz wird für das im Unterabschnitt 2.1.2 vorgestellte *Early Stopping* als Regularisierung benutzt. Der Testdatensatz stellt nach abgeschlossenem Training einen unabhängigen Test zur Verfügung, mit dem die Klassifikationsgüte der Netze bewertet wird. Für die Aufteilung der Daten auf die drei Datensätze ist es wichtig, die in Unterabschnitt 2.1.2 vorgestellte Problematik zu beachten, nach welcher eine ungleiche Klassenverteilung beim Trainieren das Netz beeinflusst. Daher werden die Daten für den Trainingsdatensatz so aufgeteilt, dass darin positive und negative Trainingsbeispiele, also je Daten der Klasse *Krater* oder *Hintergrund*, zu gleichen Anteilen vorkommen. Da die meisten Daten für das Training der Netze genutzt werden sollen, werden zunächst 80 % aller positiven Kandidaten diesem Teil zugeordnet. Die Kandidaten werden dazu zufällig aus allen positiven Beispielen gezogen; ein Kandidat kann nur je einem der drei Teildatensätze zugeordnet werden. Anschließend werden ebenso viele negative Beispiele dem Trainingsdatensatz zugeordnet. Die restlichen positiven und negativen Kandidaten werden zu je gleichen Teilen auf den Validierungs- und den Testdatensatz aufgeteilt. Die Zusammensetzung des Datensatzes ist in Tabelle 4.4 dargestellt.

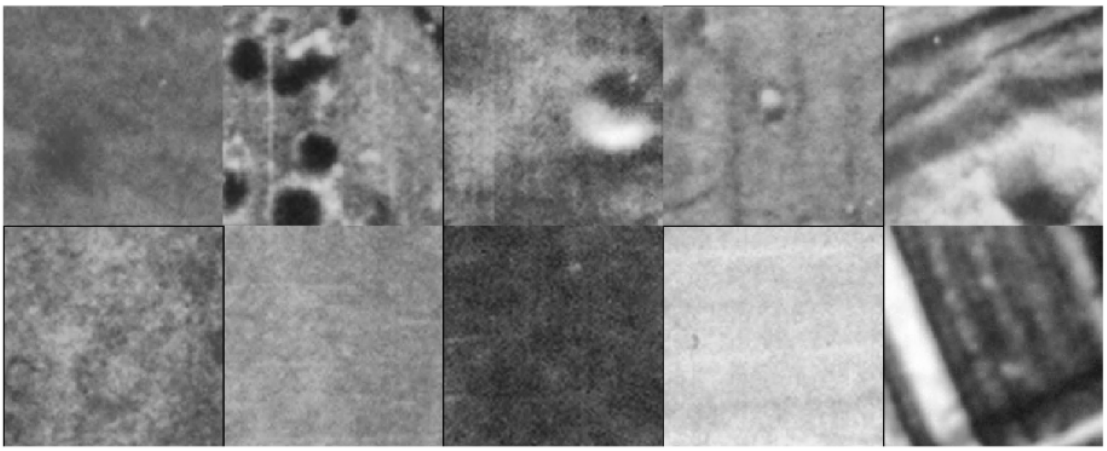

Abbildung 4.4: Positive und negative Trainingsbeispiele aus dem Datensatz

Die Abbildung zeigt in der oberen Reihe positive Trainingsbeispiele aus dem Datensatz. Da die Ausschnitte die gleichen Ausmaße in den Bildern haben, die Bilder aber einen unterschiedlichen Maßstab haben, unterscheiden sich auch die dargestellten Krater in ihrer Größe. In der unteren Reihe sind negative Trainingsbeispiele dargestellt.

	Gesamt	Training	Validierung	Test
Positive Samples	18816	15052	1882	1882
Negative Samples	411248	15052	198098	198098

Tabelle 4.4: Zusammensetzung der drei Datensätze

4.4 Neuronale Netze

In diesem Abschnitt werden die drei vorgestellten Netzvarianten experimentell untersucht. Ziel der Untersuchungen ist es, eine Parameterkonstellation für die Falsch-Positiv-Gewichtung γ und der Regularisierungsgewichtung λ_w aus Abschnitt 3.4 zu finden, welche bestmögliche Ergebnisse erzielt. Zur Bewertung der Ergebnisse werden nachfolgend geeignete Qualitätsmaße eingeführt.
Jedes tatsächlich positive und tatsächlich negative Trainingsbeispiel kann vom Netz als positiv oder negativ klassifiziert werden, woraus sich vier Fallunterscheidungen ergeben. Wird ein tatsächlich positives Trainingsbeispiel auch als positiv klassifiziert, handelt es sich um eine True-Positive-Klassifikation (TP); wird ein solches Trainingsbeispiel fälschlicherweise als negativ klassifiziert, ist es eine False-Negative-Klassifikation (FN). Beide Fälle treten auch bei tatsächlich negativen Trainingsbeispielen auf, sodass es hier die True-Negative-Klassifikation (TN) und False-Positive-Klassifikation (FP) gibt. Aus diesen Maßen lassen sich nun zwei Qualitätsmaße ableiten: So beschreibt die Vollständigkeit

$$V = \frac{TP}{TP + FN} \tag{4.1}$$

wie viele der tatsächlich positiven Trainingsbeispiele auch als solche klassifiziert wurden. Die Korrektheit

$$K = \frac{TP}{TP + FP} \tag{4.2}$$

hingegen beschreibt, wieviele der positiven Klassifikationen auch tatsächlich positive Trainingsbeispiele sind. Um die Qualität der Netze anhand eines einzelnen Maßes beschreiben zu können, wird das F1-Maß verwendet:

$$F1 = 2 \cdot \frac{V \cdot K}{V + K} \tag{4.3}$$

Dieses bildet als harmonisches Mittel aus Vollständigkeit und Korrektheit das Qualitätsmaß, anhand dessen die Netze bewertet werden können.
Das Training wird für alle Netze wie in Abschnitt 3.4 erläutert und mit jeweils 40.000 Trainingsiterationen durchgeführt. Die Parameter des *Adadelta*-Algorithmus' werden angelehnt an die Experimente in [Zeiler, 2012] zu $\rho_{ada} = 0.95$ und $\epsilon_{ada} = 1 \cdot 10^{-8}$ gewählt. Das Testen der trainierten Netze erfolgt durch die Klassifikation der Daten mit Hilfe des Testdatensatzes. Da in diesem, wie beschrieben, ein ungleiches Verhältnis zwischen positiven und negativen Testbeispielen vorliegt, wird beim Testen der spätere Anwendungsfall in der Detektion simuliert. Im Anwendungsfall treten, wie bei der Erstellung der Datensätze gezeigt, deutlich mehr Kandidaten auf, welche tatsächlich keinen Krater darstellen, als solche Kandidaten, die tatsächlich einen Kandidaten darstellen.

4.4.1 flaches CNN

Zur Untersuchung der ersten Architekturvariante werden sechs Werte für die Falsch-Positiv-Gewichtung γ zu je drei Werten für das Regularisierungsgewicht λ_w getestet, sodass insgesamt 18 Experimente durchgeführt werden. Die Ergebnisse der Experimente sind in Abbildung 4.5 dargestellt.
Wie in der Darstellung zu erkennen ist, wirkt sich eine hohe Falsch-Positiv-Gewichtung stark positiv auf die erreichten Ergebnisse aus. So wird das beste Ergebnis durch das höchste untersuchte Gewicht $\gamma = 100$ erreicht. Für die Regularisierungsgewichte lässt sich keine eindeutige Tendenz erkennen; dennoch werden die besten Ergebnisse hier meist durch $\lambda_w = 10^{-3}$ erzielt.

4.4.2 Merkmals-Extraktion

Für die Architekturvariante der Merkmals-Extraktion werden hier nur die drei besten Parameterkonstellationen aus dem vorherigen Unterabschnitt 4.4.1 untersucht. Da das Training dieser Variante im Vergleich zur vorherigen aufgrund der Tiefe des verwendeten Netzes deutlich länger dauert, werden die Experimente auf die aussichtsreichsten Konstellationen eingeschränkt.
Die Ergebnisse in Tabelle 4.5 zeigen, dass die Merkmals-Extraktion zu einer deutlichen

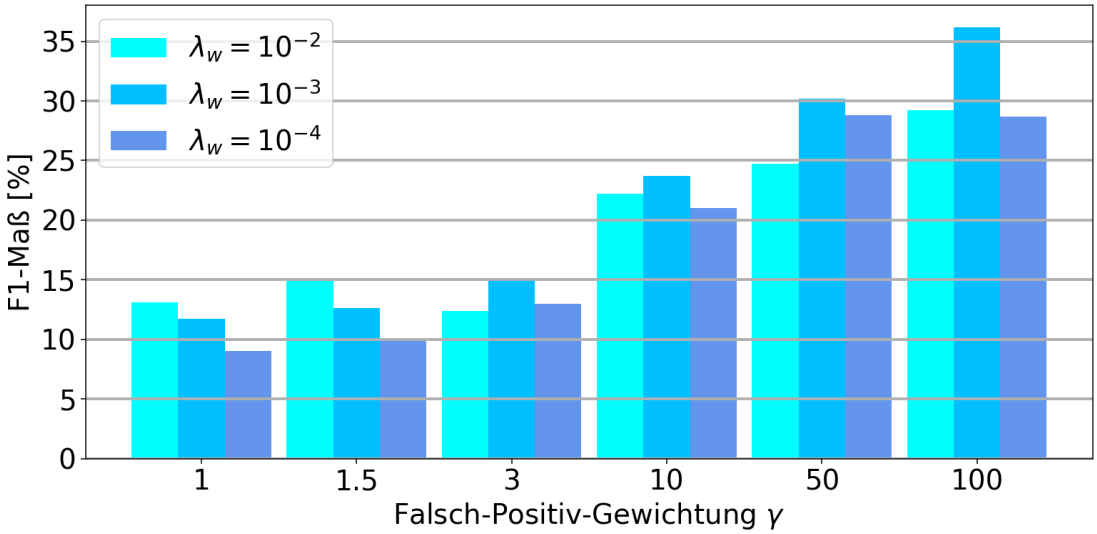

Abbildung 4.5: Ergebnisse der Untersuchungen zum flachen CNN

Das Balkendiagramm zeigt die erreichten F1-Maße für verschiedene Parameterkonstellationen aus Falsch-Positiv-Gewichtung γ und Regularisierungsgewicht λ_w.

Konstellation	**1**	**2**	**3**
γ	50	100	100
$\lambda_\mathbf{w}$	10^{-3}	10^{-2}	10^{-3}
F1-Maß [%]	31,2	45,8	37,6

Tabelle 4.5: Ergebnisse für die Merkmals-Extraktion

Verbesserung der Ergebnisse im Vergleich zum flachen CNN führt. So wird hier für die beste Parameterwahl ein F1-Maß von 45,8% erreicht, also rund 10% mehr als beim flachen CNN. Diese Ergebnisse sprechen dafür, dass die Merkmals-Extraktion durch die Nutzung eines vortrainierten Netzes besser funktioniert als das Neu-Trainieren eines flachen Netzes.

4.4.3 kombinierte Merkmals-Extraktion

Auch für die kombinierte Merkmals-Extraktion werden die drei aussichtsreichsten Varianten getestet. Die Ergebnisse sind in Tabelle 4.6 dargestellt.
Wie zu erkennen ist, kann durch die Kombination des flachen CNNs mit der Merkmals-Extraktion keine Verbesserung im Vergleich zur reinen Merkmals-Extraktion erzielt

Konstellation	1	2	3
γ	50	100	100
$\lambda_{\mathbf{w}}$	10^{-3}	10^{-2}	10^{-3}
F1-Maß [%]	31,2	41,6	25,6

Tabelle 4.6: Ergebnisse für die kombinierte Merkmals-Extraktion

werden: Die erzielten Ergebnisse sind für die beste Konstellation ca. 4 % schlechter als bei der zweiten Netzvariante. Die Vermutung, dass diese Architekturvariante durch das Lernen neuer Merkmale bessere Ergebnisse erzielt, kann damit nicht bestätigt werden. Eine mögliche Erklärung dafür ist, dass das flache CNN nicht tief genug ist, um geeignete Merkmale aus den Trainingsbeispielen zu erlernen. Stattdessen lernt es ungeeignete Merkmale, welche in Kombination mit den extrahierten Merkmalen aus dem *Inception ResNet V2* die Klassifikation negativ beeinflussen.

4.5 Detektion von Kratern

Als Abschluss dieses Kapitels wird die Detektion von Bombenkratern durch das Zusammenspiel von Krater-Kandidaten-Auswahl und Klassifikation experimentell untersucht. Die Untersuchungen für den Detektor werden, wie in Abschnitt 4.1 beschrieben, mit Hilfe von vier Luftbildern durchgeführt, die nicht für die bisherigen Experimente verwendet wurden. Insbesondere tauchen aus diesen Bildern keine Ausschnitte in den Datensätzen für die CNNs auf. In diesen Experimenten wird der Einfluss der beiden Parameter ϵ_{det} und ϵ_{IoU} auf die Ergebnisse der Detektion untersucht. Analog zu Abschnitt 4.2 wird eine nach beiden Filterungsschritten noch vorhandene Detektion dann als korrekt gewertet, wenn der Abstand zu einer Kraterreferenz kleiner ist als der gewählte Schwellwert von $\epsilon_{blob} = 80$ px. Auch hier kann eine Detektion nur einer Referenz zugeordnet werden; andere Referenzen, die das Abstandskriterium erfüllen, müssen dann anderen Detektionen zugeordnet werden. Die Experimente für die Detektion werden mit der besten Variante des Blob-Detektors (s. Tabelle 4.3, Versuch 13) sowie der besten Variante der Klassifikation (s. Tabelle 4.5, Konstellation 2) durchgeführt, da durch diese Wahl die beste Detektionsgenauigkeit erreicht werden kann. In den Abbildungen 4.6 bis 4.9 sind die Ergebnisse der Detektionsexperimente für die vier Testbilder dargestellt.

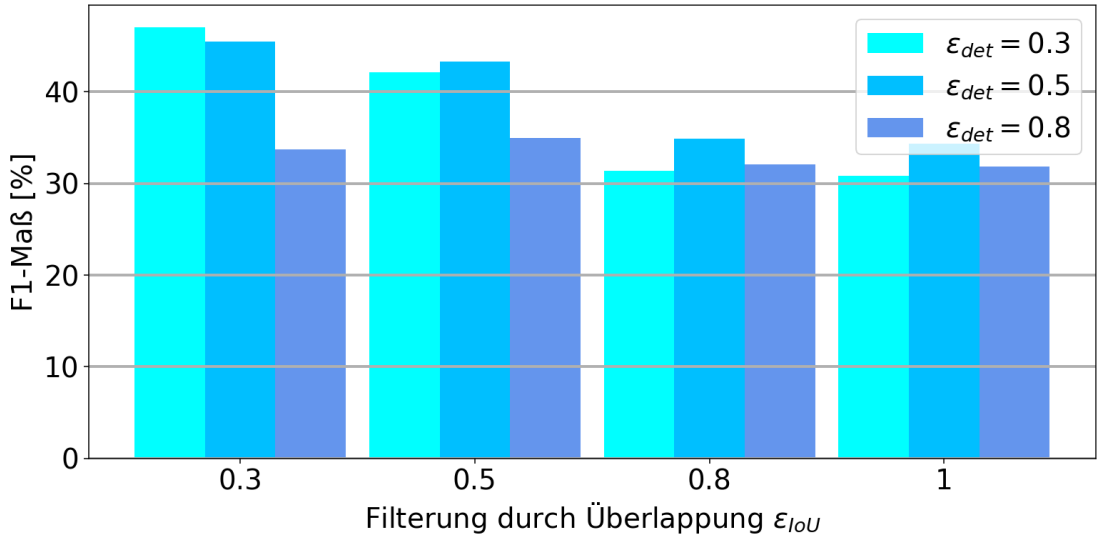

Abbildung 4.6: F1-Maße für Detektionen im 1. Testbild

In dem Diagramm sind die Ergebnisse der Experimente für das 1. Testbild dargestellt. Zusammenstehende Balken stellen Ergebnisse für einen gleichen Überlappungsschwellwert ϵ_{IoU} dar. Balken gleicher Farbe stellen Ergebnisse für einen gleichen Detektionsschwellwert ϵ_{det} dar. Die Ergebnisse zeigen, dass in diesem Testbild ein kleinerer Überlappungsschwellwert ϵ_{IoU} zu besseren Ergebnissen führt. Das beste F1-Maß von 47,1 % wird erreicht, wenn beide Parameter auf $0{,}3$ gesetzt werden.

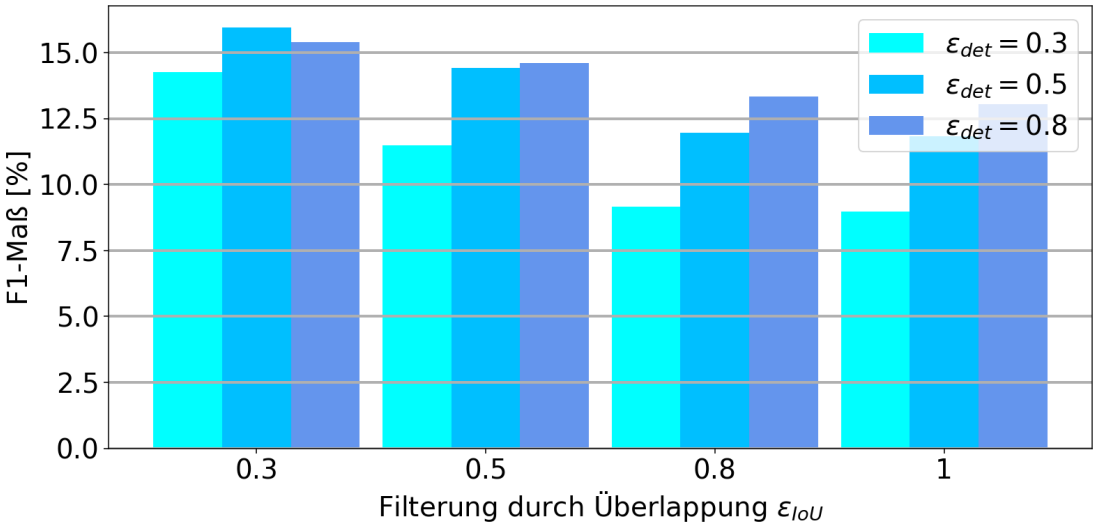

Abbildung 4.7: F1-Maße für Detektionen im 2. Testbild

In dem Diagramm sind die Ergebnisse der Experimente für das 2. Testbild dargestellt. Zusammenstehende Balken stellen Ergebnisse für einen gleichen Überlappungsschwellwert ϵ_{IoU} dar. Balken gleicher Farbe stellen Ergebnisse für einen gleichen Detektionsschwellwert ϵ_{det} dar. Die Ergebnisse zeigen auch hier, dass ein kleinerer Überlappungsschwellwert ϵ_{IoU} zu besseren Ergebnissen führt. Zudem lassen die Ergebnisse die Tendenz erkennen, dass hier ein größerer Detektionsschwellwert zu besseren Ergebnissen führt. Das beste F1-Maß von 16,0 % wird für $\epsilon_{IoU} = 0{,}3$ und $\epsilon_{det} = 0{,}5$ erreicht.

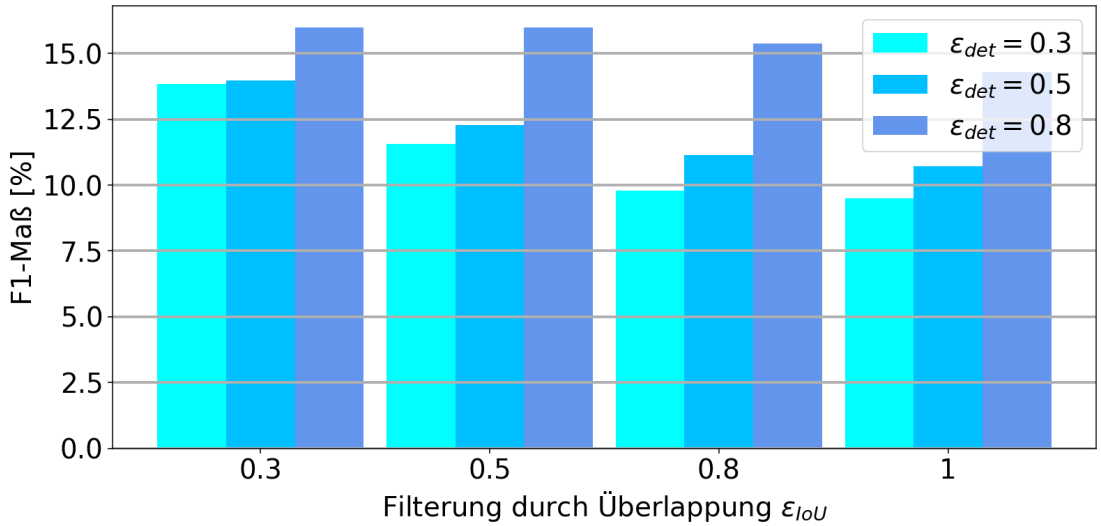

Abbildung 4.8: F1-Maße für Detektionen im 3. Testbild

In dem Diagramm sind die Ergebnisse der Experimente für das 3. Testbild dargestellt. Zusammenstehende Balken stellen Ergebnisse für einen gleichen Überlappungsschwellwert ϵ_{IoU} dar. Balken gleicher Farbe stellen Ergebnisse für einen gleichen Detektionsschwellwert ϵ_{det} dar. Die Ergebnisse zeigen auch hier, dass ein kleinerer Überlappungsschwellwert ϵ_{IoU} zu besseren Ergebnissen führt. Zudem lassen auch hier die Ergebnisse die Tendenz erkennen, dass ein größerer Detektionsschwellwert zu besseren Ergebnissen führt. Das beste F1-Maß von 16,0 % wird für $\epsilon_{IoU} = 0{,}3$ und sowohl $\epsilon_{det} = 0{,}5$ als auch$\epsilon_{det} = 0{,}8$ erreicht.

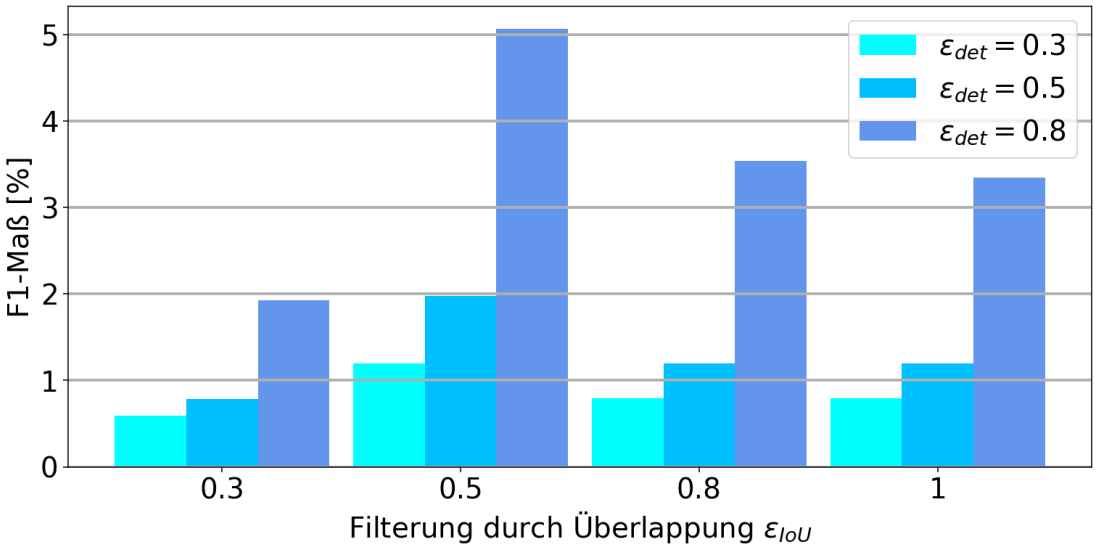

Abbildung 4.9: F1-Maße für Detektionen im 4. Testbild

In dem Diagramm sind die Ergebnisse der Experimente für das 4. Testbild dargestellt. Zusammenstehende Balken stellen Ergebnisse für einen gleichen Überlappungsschwellwert ϵ_{IoU} dar. Balken gleicher Farbe stellen Ergebnisse für einen gleichen Detektionsschwellwert ϵ_{det} dar. Die Ergebnisse lassen keine klare Aussage zur Auswirkung des Überlappungsschwellwerts zu. Allerdings lassen auch hier die Ergebnisse die Tendenz erkennen, dass ein größerer Detektionsschwellwert zu besseren Ergebnissen führt. Das beste F1-Maß von 5,1 % wird für $\epsilon_{IoU} = 0{,}5$ und $\epsilon_{det} = 0{,}8$ erreicht.

Die großen Unterschiede zwischen den Ergebnissen für die vier Testbilder lassen sich damit erklären, dass in den Bildern unterschiedlich viele Krater enthalten sind: Wie in Abschnitt 4.1 erwähnt, sind im ersten Testbild 801, im zweiten 18, im dritten 16 und im letzten 4 Krater enthalten. Die Anzahl der Krater-Kandidaten hängt allerdings wenig von der tatsächlichen Anzahl der Krater ab: Durch die gewählten Parameter des Blob-Detektors werden zum Beispiel auch Häuser, Bäume oder deren Schattenwürfe als Blobs und damit als Kandidaten extrahiert. Die Anzahl der Kandidaten liegt zwischen 25.000 und 40.000 pro Bild. Damit liegt in den letzten drei Bildern ein deutlich geringeres Verhältnis von positiven zu negativen Testbeispielen vor. Demnach fallen hier die Ergebnisse schlechter aus, weil dieses Verhältnis nicht jenem entspricht, welches durch die Falsch-Positiv-Gewichtung beim Training der Netze angenommen wurde. Das Verhältnis im ersten Testbild entspricht eher dem Szenario im Training. Ebenso haben mehr False-Positive-Detektionen einen stärkeren negativen Einfluss auf die Korrektheit, wenn es nur wenige True-Positive-Detektionen gibt. Analog dazu wird die Vollständigkeit bei einer False-Negative-Detektion deutlich schlechter, wenn es nur wenige True-Positive-Detektionen gibt. Diese Umstände führen dazu, dass im ersten Testbild deutlich bessere Ergebnisse erzielt werden als in den anderen drei Testbildern.

Das Verhältnis von positiven und negativen Testbeispielen in den letzten drei Bildern erklärt auch die Tendenz, dass ein höherer Detektionsschwellwert ϵ_{det} die Ergebnisse verbessert: Ist dieser Wert höher, werden durch den Filterschritt mehr Detektionen verworfen. Da in diesen Bildern deutlich mehr negative als positive Testbeispiele vorliegen, werden auch mehr negative Kandidaten verworfen, was die Korrektheit und damit das F1-Maß steigert.

KAPITEL 5

Fazit und Ausblick

In dieser Arbeit wurde ein Verfahren erarbeitet, welches durch die Nutzung von Convolutional Neural Networks die Detektion von Bombenkratern in historischen Luftbildern ermöglicht.
Am Anfang des Verfahrens steht die Krater-Kandidaten-Auswahl, realisiert durch einen Blob-Detektor. Hier kann eine sehr hohe Vollständigkeit erreicht werden, wie es auch die selbst festgelegte primäre Anforderung an diesen Schritt ist. Allerdings führt die Umsetzung der Anforderung auch dazu, dass übermäßig viele Kandidaten extrahiert werden, die tatsächlich keinen Krater darstellen. Um diesem Problem entgegenzuwirken, könnte die Anforderung auch die Korrektheit der Kandidaten-Extraktion berücksichtigen; die Untersuchungen beim Blob-Detektor haben gezeigt, dass eine Vollständigkeit von über 95 % möglich ist, während die Anzahl der Blobs um ein Drittel reduziert wird (vgl. Tabelle 4.2, Versuch 7).

Im Anschluss an die Krater-Kandidaten-Auswahl wurde die Erstellung der Datensätze für die CNNs vorgestellt. Indem ein Ausschnitt fester Größe um einen Blob gebildet wurde, konnte aus jedem Blob ein Datenbeispiel bzw. Sample für die Netze gebildet werden. Dieses Vorgehen führt dazu, dass in einem Ausschnitt sowohl kleine als

auch große, sowohl viele als auch einzelne Krater vorhanden sein können. Dies stellt ein Problem dar, da dadurch die Bildausschnitte für die Klasse *Krater* sehr stark variieren, wodurch den Netzen die Klassifikation erschwert wird. Dieses Problem könnte dadurch behoben werden, dass die Größe der Blobs bei der Erstellung der Samples berücksichtigt wird, sodass folglich die Größe des Ausschnitts der Größe des Blobs entspricht. Für die Vergabe der Klassenzugehörigkeiten der Ausschnitte muss dann neben der Position auch die Größe eines Ausschnitts mit der Größe der Referenzkrater verglichen werden. Dafür käme als geeignetes Maß die *Intersection over Union* in Frage. Hier besteht allerdings das Problem, dass die Referenzkrater sich in Position und Radius nur indirekt auf die tatsächlichen Krater beziehen (vgl. Abschnitt 4.1). Dadurch kann die Klassenzugehörigkeit durch Vergleich von Position und Größe nicht zuverlässig korrekt bestimmt werden. An dieser Stelle ist entweder eine Aufbereitung der Referenzdaten nötig, sodass die Referenzen nur den inneren Gebieten der Krater entsprechen, den Kraterauswurf also nicht beinhalten. Alternativ ist es möglich, die Größe der Referenzen und Blobs bei der Vergabe der Klassenzugehörigkeiten nicht zu beachten, sodass diese wie beschrieben nur anhand der Distanz festgelegt wird. Dies hätte zur Folge, dass im Datensatz teilweise fehlerhafte Samples auftauchen, deren Klassenzugehörigkeiten also teilweise nicht korrekt sind. Ein weiteres Problem, dass sich aus den festen Ausschnittgrößen im Datensatz ergibt, ist, dass die gebildeten Samples sich teilweise überschneiden können. Somit können Teile der Bildausschnitte sowohl im Trainings- als auch im Validierungs- und Testdatensatz auftauchen. Dies sollte vermieden werden, damit die letzteren beiden Datensätze eine wirklich unabhängige Kontrolle für das Training darstellen.
Bei der Aufteilung aller vorhandenen Daten auf die drei Teildatensätze wurden dem Trainingsdatensatz gleich viele positive wie negative Samples zugewiesen. Dieses Vorgehen wurde mit dem Hinweis auf den in Unterabschnitt 2.1.2 vorgestellten Zusammenhang gewählt, dass ein neuronales Netz mit gleich vielen Samples aller Klassen trainieren sollte. Diese Anforderung wird vor allem dadurch eingehalten, dass in jedem Minibatch gleich viele positive wie negative Samples auftreten; im gesamten Trainingsdatensatz kann die Verteilung damit trotzdem ungleich sein. Der Vorteil davon wäre, dass mehr verschiedene Samples der Klasse *Hintergrund* im Training zur Verfügung stünden. Da diese Klasse deutlich mehr Variationen aufweist, indem sie z.B.

Bäume oder Häuser beinhaltet, wären diese Variationen dadurch besser im Training repräsentiert. Dies könnte die Klassifikationsgenauigkeiten für diese Klasse und damit auch die Ergebnisse der Kraterdetektion verbessern. In Bezug auf die Anwendung des entwickelten Verfahrens zur Detektion von Bombenkratern in bisher unbekannten Luftbilder ist es sinnvoll, die Auflösung der Luftbilder als Unbekannte zu eliminieren. Dazu könnten alle verwendeten Bilder auf eine gemeinsame Bodenpixelgröße skaliert werden. Allerdings stellen sich auch hier die Probleme, dass eine feste Ausschnittgröße dazu führt, dass die Krater die Ausschnitte unterschiedlich ausfüllen, wohingegen eine der Blobgröße angepasste Ausschnittgröße durch die gegebenen Referenzkrater behindert wird.

Im Arbeitsablauf folgte auf die Erstellung der Datensätze die Auswahl der CNN-Architekturen. Hier konnten durch die Variante der Merkmals-Extraktion die besten Ergebnisse erzielt werden. Die Unterschiede in den Ergebnissen zum flachen CNN machen deutlich, dass das Lernen neuer, weniger komplexer Merkmale der Nutzung von bereits gelernten Merkmalen unterliegt. Die Annahme, dass ein flaches CNN geeignete Merkmale neu lernen kann, konnte damit für die vorliegenden Daten nicht bestätigt werden. Die kombinierte Merkmalsextraktion hat somit auch keine Verbesserung der Klassifikation erreichen können. Dies könnte darauf zurückzuführen sein, dass das flache CNN von selbst keine geeigneten Merkmale lernen kann. Lernt es stattdessen ungeeignete Merkmale, wird dadurch die Klassifikation anhand des kombinierten Merkmalsvektor behindert. Dieses Problem könnte dadurch behoben werden, dass das flache CNN in seiner Tiefe erweitert wird, um geeignetere Merkmale zu lernen. Dies würde allerdings auch die Anzahl der zu lernenden Parameter und damit die Menge der benötigten Trainingsdaten vergrößern.

Anschließend an die Gestaltung der Architekturen der CNNs folgte die Untersuchung, wie sich die modifizierte Verlustfunktion auf das Training auswirkt. Hier konnte gezeigt werden, dass die Verwendung eines Gewichtungsparameters für den falsch-positiven Anteil der Verlustfunktion zu einer deutlichen Verbesserung der Ergebnisse führt. Allerdings kann diese Verbesserung der Ergebnisse dadurch erklärt werden, dass diejenige Gewichtung, die die besten Ergebnisse erzielt, dem Verhältnis von

positiven zu negativen Samples im Testdatensatz entspricht. Obwohl es also durch den Gewichtungsparameter möglich ist, dieses Verhältnis zu repräsentieren, so ist doch das Verhältnis von positiven und negativen Kandidaten in jedem Bild unterschiedlich. Die Wahl des Gewichtungsparameters kann also nicht sowohl für Luftbilder mit vielen Kratern wie auch für Luftbilder mit wenigen Kratern optimal getroffen werden.

Den Abschluss der entwickelten Methodik stellt die Detektion dar. Hier wurden zwei Filterungsschritte vorgestellt, durch welche es möglich war, die Krater-Kandidaten abhängig von prädizierter Klassenzugehörigkeit und jeweiliger Überlappung zu filtern. Auch hier wurde die Problematik deutlich, die im vorherigen Absatz benannt wurde: Je nach dem, wie hoch oder niedrig die Belastung in einem Luftbild ist, hat die Wahl der Filterparameter unterschiedliche Auswirkungen. Hier konnten keine global optimalen Parameter gefunden werden. Diese Problematik könnte zumindest für die Wahl des Überlappungsparameters dadurch reduziert werden, dass die durch die Krater-Kandidaten-Auswahl gegebenen Ausschnitte nicht gleich groß, sondern in ihrer Größe an die der Krater angepasst wären. Dadurch könnten Überlappungen zwischen Detektionen auf solche Fälle reduziert werden, in denen sich die detektierten Krater auch tatsächlich überlappen. Damit ließe sich die Filterung in der Detektion auf den Detektionsschwellwert beschränken.

Alternative Lösungsansätze ergeben sich, wenn das vorliegende Problem nicht als eine reine Detektionsaufgabe verstanden wird. Wie in Abschnitt 1.2 vorgestellt, gibt es Verfahren, die sich auf die Erstellung einer Belastungskarte konzentrieren. Hierbei geht es nicht primär um die Detektion einzelner Krater, sondern um das Feststellen eines Bereichs, der durch Bombenkrater belastet ist. Auch diese Problemstellung kann durch neuronale Netze angegangen werden, so zum Beispiel durch *Fully Convolutional Networks* [Long et al., 2015]. Diese Art von neuronalen Netzen prädiziert die Klasse nicht für das gesamte Eingangsbild, sondern für jedes Pixel des Eingangsbildes. Damit kann ein gegebenes Luftbild in belastete und nicht-belastete Bereiche segmentiert werden. Dieser Lösungsansatz würde einen Großteil der Probleme umgehen, die sich bei der Detektion, insbesondere bei der Krater-Kandidaten-Auswahl und der Größe der Ausschnitte, ergeben haben. Die Voraussetzung für ein solches Vorgehen wäre, dass

die Belastungskarte für jedes Luftbild bekannt ist, welches zum Training genutzt wird. Diese Belastungskarten können, wie in [Kruse et al., 2018] beschrieben, mit Hilfe von Expertenwissen aus den hier gegebenen Referenzkratern generiert werden.

Zusammenfassend können die in dieser Arbeit entwickelten Methoden dafür genutzt werden, Bombenkrater in historischen Luftbildern zu detektieren. Die Ergebnisse sprechen allerdings dagegen, dass sich das entwickelte Verfahren für eine rein automatische Detektion eignet. Da weder alle Bombenkrater im Bild gefunden werden, noch alle Detektionen tatsächlich Bombenkrater darstellen, bedarf es bei der Verwendung des entwickelten Verfahrens menschlicher Kontrolle. Dennoch ist es möglich, die manuelle Krater-Detektion durch das vorgestellte Verfahren zu unterstützen.

Literaturverzeichnis

Literatur

[Bishop, 2006] BISHOP, Christopher M. Pattern Recognition and Machine Learning, Springer, New York, 2006.

[Bradski, 2000] BRADSKI, Gary; KAEHLER, Adrian. OpenCV. Dr. Dobb's journal of software tools, 2000, 3. Jg.

[Brenner et al., 2018] BRENNER, Simon; ZAMBANINI, Sebastian; SABLATNIG, Robert. Detection of Bomb Craters in WWII Aerial Images. In: Proceedings of the OAGM Workshop. 2018. S. 94-97.

[Cohen et al., 2016] COHEN, Joseph Paul; LO, Henry Z.; LU, Tingting; DING, Wei. Crater detection via convolutional neural networks. arXiv preprint arXiv:1601.00978, 2016.

[Duda et al., 2012] DUDA, Richard O.; HART, Peter E.; STORK, David G. Pattern classification. John Wiley & Sons, 2012.

[Emami et al., 2015] EMAMI, Ebrahim; BEBIS, George; NEFIAN, Ara; FONG, Terry. Automatic crater detection using convex grouping and convolutional neural

networks. In: International Symposium on Visual Computing. Springer, Cham, 2015. S. 213-224.

[Glorot & Bengio, 2015] GLOROT, Xavier; BENGIO, Yoshua. Understanding the difficulty of training deep feedforward neural networks. In: Proceedings of the thirteenth international conference on artificial intelligence and statistics. 2010. S. 249-256.

[He et al., 2015] HE, Kaiming; ZHANG, Xiangyu; REN, Shaoqing; SUN, Jian. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In: Proceedings of the IEEE international conference on computer vision. 2015. S. 1026-1034.

[Huang et al., 2016] HUANG, Gao; LIU, Zhuang; VAN DER MAATEN, Laurens; WEINBERGER, Kilian Q. Densely connected convolutional networks. In: CoRR, vol. abs/1608.06993, 2016.

[Kruse et al., 2018] KRUSE, Christian; ROTTENSTEINER, F.; HOBERG, T.; ZIEMS, M.; REBKE, J.; HEIPKE, C.. Generating impact maps from automatically detected bomb craters in aerial wartime images using marked point processes. In: ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences 4 (2018), 4. (3), 2018, S. 127-134.

[Le Cun et al., 1989] LECUN, Yann; BOTTOU, Léon; BENGIO, Yoshua; HAFFNER, Patrick. Gradient-based learning applied to document recognition. In: Proceedings of the IEEE, 1998, 86. (11), S. 2278-2324.

[Long et al., 2015] LONG, Jonathan; SHELHAMER, Evan; DARRELL, Trevor. Fully convolutional networks for semantic segmentation. In: Proceedings of the IEEE conference on computer vision and pattern recognition. 2015. S. 3431-3440.

[Ruoff, 2013] RUOFF, Manuel: Nicht immer hat's gekracht. In: Preußische Allgemeine Zeitung vom 14. Dezember 2013, S. 4.

[Meng et al., 2013] DING, Meng; CAO, Yunfeng; WU, Qingxian. Novel approach of crater detection by crater candidate region selection and matrix-pattern-oriented

least squares support vector machine. In: Chinese Journal of Aeronautics, 2013, 26. (2), S. 385-393.

[Merler et al., 2005] MERLER, Stefano; FURLANELLO, Cesare; JURMAN, Giuseppe. Machine learning on historic air photographs for mapping risk of unexploded bombs. In: International Conference on Image Analysis and Processing. Springer, Berlin, Heidelberg, 2005. S. 735-742.

[Russakovsky et al., 2015] RUSSAKOVSKY, Olga, et al. Imagenet large scale visual recognition challenge. International Journal of Computer Vision, 2015, 115. (3), S. 211-252.

[Simard et al., 2003] SIMARD, Patrice Y.; STEINKRAUS, Dave; PLATT, John C. Best practices for convolutional neural networks applied to visual document analysis. In: IEEE, 2003. S. 958.

[Szegedy et al., 2017] SZEGEDY, Christian; IOFFE, Sergey; VANHOUCKE, Vincent; ALEMI, Alexander. Inception-v4, inception-resnet and the impact of residual connections on learning. In: 31. AAAI. 2017. S. 12.

[Urbach & Stepinksi, 2009] URBACH, Erik R.; STEPINSKI, Tomasz F. Automatic detection of sub-km craters in high resolution planetary images. In: Planetary and Space Science, 2009, 57. (7), S. 880-887.

[Zeiler, 2012] ZEILER, Matthew D. ADADELTA: an adaptive learning rate method. arXiv preprint arXiv:1212.5701, 2012.

[Zell, 1994] ZELL, Andreas. Simulation neuronaler Netze. Bonn: Addison-Wesley, 1994.

Internet-Quellen

[Embedded Vision Alliance, 2019] Using Convolutional Neural Networks for Image Recognition. `https://www.embedded-vision.com/platinum-members/cadence/embedded-vision-training/documents/`

pages/neuralnetworksimagerecognition
(Letzter Aufruf: 16.01.2019)

[gubd.de, 2018] Luftbildauswertung: Kampfmittelvorerkundung.
https://gubd.de/geo/luftbildauswertung/kampfmittel
(Letzter Aufruf: 29.01.2019)

[PetarV, 2016] 2D-Convolutions.
https://github.com/PetarV-/TikZ/tree/master/2D%20Convolution
(Letzter Aufruf: 16.01.2019)

[Prabhu, 2018] Understanding of Convolutional Neural Network (CNN) - Deep Learning.
https://medium.com/@RaghavPrabhu/understanding-of-convolutional-neural-network-cnn-deep-learning-99760835f148
(Letzter Aufruf: 16.01.2019)

Danksagung

Ich möchte an dieser Stelle dem Landesamt für Geoinformation und Landesvermessung Niedersachsen und dem Kampfmittelbeseitigungsdienst Niedersachsen der Regionaldirektion Hameln-Hannover dafür danken, dass sie die Daten für diese Arbeit zur Verfügung gestellt haben.