Kapitel 1

Methoden

1.1 Trainingsdaten

Abbildung 1.1: RGB-Sentinel-2-Aufnahme des zu untersuchenden Ackers. Die infizierten Flächen sind weiß umrandet.

Der infizierte Acker, der die Basis des Datensatzes bildet, befindet sich etwa 15 km nordwestlich von Bologna in Norditalien ($44°34'28.92''$ Nord, $11°10'21.36''$ Ost). Das Feld hat eine Fläche von $7640,57$ m^2 und ist mit Sorghum bepflanzt. Mitarbeiter des CREA (Council for Agricultural Research and Economics) haben vor Ort am 12.07.2018 Befälle von Anthracnose und der bakteriellen Streifenkrankheit (med.: Xanthomonas translucens) diagnostiziert. Etwa die Hälfte der Pflanzen des Feldes sind betroffen. Dabei sind im östlichen Teil des Feldes (Abb. 1.1, innere Markierung) auf einer Fläche von $1043,22$ m^2 von etwa 60 bis 70% der Pflanzen befallen.

1.2 Normalized Difference Vegetation Index

Es gibt eine starke Korrelation zwischen dem physiologischen Status einer Pflanze und deren Chlorophyllgehalt. Faktoren wie Krankheit, Dürre oder Umweltverschmutzung haben einen negativen Einfluss auf den Chlorophyllspiegel.[12] Messungen haben ergeben, dass es einen Verbindung zwischen dem Reflexionsgrad im nahen Infrarotbereich und im Rotbereich und dem Chlorophyllgehalt gibt. Das heißt, dass eine gesunde, adulte Pflanze im nahen Infrarotbereich stärker reflektiert als zum Beispiel eine pathologisch veränderte Pflanze. Jedoch bleibt die Reflexion im roten Lichtspektrum in beiden Fällen vergleichsweise schwach. Andere vegetationsfreie Oberflächen wie Acker, Straßen oder Wasser strahlen auch im nahen Infrarotbereich schwach zurück. Dadurch ergibt sich eine zerstörungsfreie Methode, mit einer Multispektralkamera die Vitalität („Grünheit“) einer oder mehrerer Pflanzen zu bestimmen.[10]

Eine multispektralen Aufnahme kann mithilfe der Formel

$$NDVI = \frac{Band_{NIR} - Band_{Red}}{Band_{NIR} + Band_{Red}} \tag{1.1}$$

dazu genutzt werden, den *Normalized Difference Vegetation Index* (NDVI) zu berechnen. Wobei $Band_{NIR}$ der nahe Infrarotbereich (Near Infrared) und $Band_{RED}$ der sichtbare rote Bereich des elektromagnetischen Spektrums ist. Der NDVI gibt quantifizierte Werte im Bereich von -1 bis 1 zurück. Dabei deuten Werte, die kleiner als 0 sind, auf Wasserobflächen hin. 0 bedeutet keine Vegetation. Bei Werte nahe 0 handelt es sich um spärliche oder ungesunde Vegetation. Das bedeutet je näher ein Wert an 1 ist, desto dichter bewachsen und gesünder ist die beobachtete Vegetationsfläche.[15] Dass bei einem niedrigen, positiven NDVI nicht unterschieden werden kann, ob eine Fläche kaum bewachsen ist oder ungesunde Vegetation besitzt, kann hier vernachlässigt werden. Das Gebiet, das in dieser Arbeit untersucht wird, ist ein bewachsenes Feld, so kann man geringe Vegetation ausschließen.

1.3 Sentinel-2

Die Sentinel-2-Satelliten sind eine von sechs Satellitenarten (Sentinel-1 bis -6) des Copernicus-Programms[1], die zur Erdbeobachtung in einen 786 km ho-

[1] Das Copernicus-Programm wurde von der Europäischen Union zur Erdbeobachtung ins Leben gerufen. Die gesammelten Daten werden für wissenschaftliche, wirtschaftliche und

hen sonnensynchronen Orbit gebracht wurden. Die Instrumente der Sentinel-2-Satelliten können Aufnahmen in Bereichen des roten und nahen Infrarot- bis hin zum Kurzwelleninfrarotspektrums. Die Aufnahmen haben Gesamtgröße von $100 * 100$ km und je nach Band eine von Auflösung von 10m, 20m oder 60m (s. Tabelle 1.1).

Bandnummer	Räumliche Auflösung	Mittlere Wellenlänge (nm)	Bandbreite (nm)
B1	60	443,9	27
B2	10	496,6	98
B3	10	560	45
B4	10	664,5	38
B5	20	703,9	19
B6	20	740,2	18
B7	20	782,5	28
B8	10	835,1	145
B8a	20	864,8	33
B9	60	945	26
B10	60	1373,5	75
B11	20	1613,7	143
B12	20	2202,4	242

Tabelle 1.1: Räumliche und spektrale Auflösungen von Sentinel-2A[8]

Besonders wichtig sind die Bänder B4 (Rot) und B8 (Nahes Infrarot). Mit diesen Bändern kann der NDVI (s. Kapitel 1.2) berechnet werden.[7] Die Sentinel-2-Satelliten bieten mit $10 * 10$ m pro Pixel eine hohe räumliche Auflösung.[2] Diese Eigenschaft ist wichtig, um eine mögliche Infizierung genau eingrenzen zu können.

Dabei ist es auch wichtig, dass die Satelliten regelmäßige Daten liefern können. Durch die gemeinsame Konstellation übertragen die Plattformen alle fünf Tage Daten über einen spezifischen Punkt auf der Erdoberfläche.[9] Damit ist gewährleistet, dass der Feldbesitzer ohne persönliche Inspektion ein bis zweimal in der Woche eine Gesundheitseinschätzung über seine Felder erhält.

private Anwendungszwecke zur Verfügung gestellt.[5]

[2] Im Vergleich hat zum Beispiel der Landsat-8-Satellit, dessen Daten ebenfalls frei verfügbar sind, eine relativ geringe Auflösung von $30 * 30$ m.[18]

1.4 Mask R-CNN

In Kapitel 1.2 und 1.3 wurde erklärt wie Daten über die möglichen Erkrankungen geliefert und verarbeitet werden können. Auf den zugrunde liegenden Bilddaten soll nun ein künstliches neuronales Netzwerk (KNN) trainiert werden. In diesem Kapitel wird darauf eingegangen, welche Anforderungen an das KNN gestellt werden, warum das Titel gebende Netz ausgewählt wurde und wie dieses funktioniert.

1.4.1 Anforderungen

Das KNN muss in der Lage sein, wahrscheinliche Krankheiten in der zu untersuchenden Agrarfläche möglichst genau eingrenzen und klassifizieren zu können. Das ist besonders wichtig, wenn ein Feld von multiplen Krankheiten betroffen ist.

Es ist damit zu rechnen, dass Daten unter bewölkten Bedingungen aufgenommen werden. Nach starken Niederschlägen können Acker teils oder gänzlich überflutet sein.[14] Das sorgt selbst unter wolkenfreien Bedingungen für einen niedrigen NDVI, obwohl die Nutzpflanzen gesund sind. Das neuronale Netz muss mit solchen „Ausreißern“ umgehen können.

Daraus ergeben sich folgende Kriterien für das neuronale Netzwerk:

- Erkennung auf Pixelebene
- Robustheit
- Hohe Genauigkeit

1.4.2 Grundlagen

Vollständig vernetztes neuronales Netz

Künstliche neuronale Netze sind mathematische Modelle, die nach dem Vorbild von biologischen neuronalen Netzen gebildet worden sind. So ist ein KNN ebenfalls eine Verbindung von künstlichen Neuronen. Diese Neuronen sind in Schichten angeordnet und jede die Neuronen einer Schicht sind mit den Neuronen nächsten bzw. letzten Schicht verbunden. Zwischen der ersten und der

letzten sog. Ausgangsschicht existieren n versteckte Schichten (engl.: hidden layers).

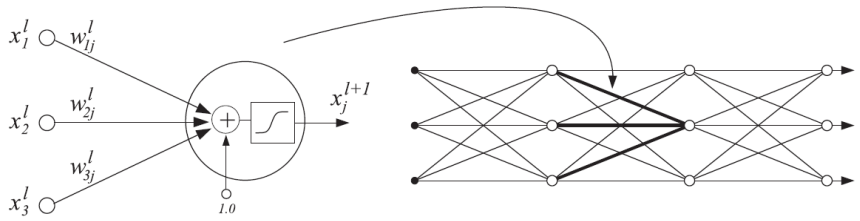

Abbildung 1.2: Künstliches neuronales Netz[19]

Ein Neuron besitzt mehrere Eingangsverbindungen (Gewichte) und ein Ausgangsneuron. Ob ein Neuron „feuert“, wird durch eine lineare oder nichtlineare Aktivierungsfunktion bestimmt. Die Eingangsgewichte sind veränderbare Werte, die je nach Höhe einen starken oder niedrigen Einfluss auf die Aktivierungsfunktion haben.

$$x_j^{l+1} = f(\sum_i w_{ij}^l x_i^l + w_{bj}^l) \tag{1.2}$$

beschreibt das Neuron j in Schicht $l + 1$, wobei

- w_{ij}^l die Gewichte sind, die Neuron i in Schicht l mit Neuron j verbinden.
- w_{bj}^l der Biasterm des j-ten Neurons in Schicht l ist.
- f die Aktivierungsfunktion ist.[19]

Convolutional Neural Networks

Convolutional Neural Networks (CNN, dt.: faltendes neuronales Netzwerk) sind Kategorien von neuronalen Netzen, die besonders in der *Computer Vision* Anwendung finden. In der ersten Schicht werden mehrere Merkmale (engl.: features) durch Filter extrahiert und in separate sog. *Feature Maps* abgelegt, um größere Abstraktionsebenen zu erreichen. Diese Filter sind mathematisch mit Faltungen (engl.: convolutions) zu vergleichen und geben dem Netz den Namen.

Die Dimensionen der Feature Maps werden in einem Poolingschritt[3] (oder

[3]Es gibt verschiedene Arten von Pooling (Max, Average, Sum, ...). Dabei wird die $m * m$ px große Feature Map in sich angrenzende $n * n$ px große Felder eingeteilt ($n < m$). Im Falle von Max-Pooling wird der höchste Wert aus dem Feld übernommen.

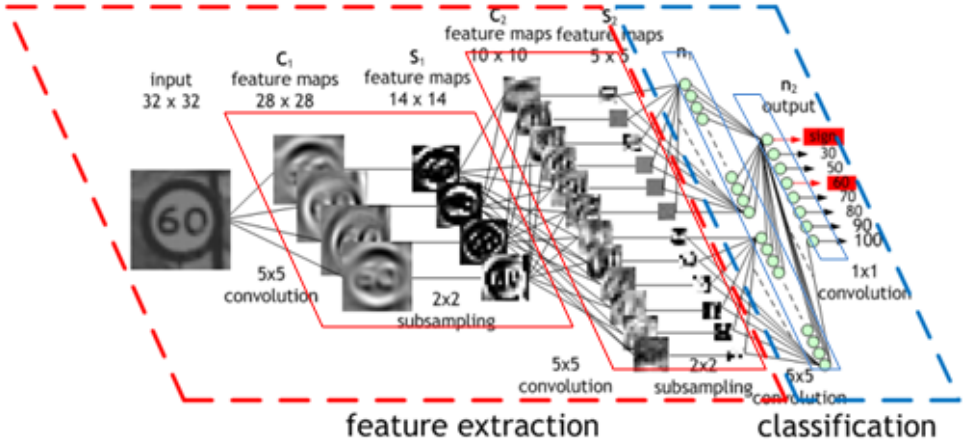

Abbildung 1.3: Architektur eines Convolutional Neural Network[6]

auch *subsampling*) reduziert. Dadurch bleiben nur relevante Informationen erhalten und das CNN wird bis zu einem gewissen Grad robust gegenüber Translationen und Rotationen. In der Regel werden die Faltungen und das das Pooling zwei Mal durchgeführt, wie es in Abb. 1.3 abgebildet ist.

Nach der Merkmalextraktion werden die Feature Maps zur Klassifikation in eine eindimensionale Schichten geglättet. Die folgenden Schichten bis zur Ausgangsschicht sind vollständig vernetzt.

1.4.3 Mask R-CNN

Im Rahmen dieser Arbeit wird das *Mask Region-based Convolutional Neural Network* untersucht. Mask R-CNN ist eine von Facebook AI Research (FAIR) entwickelte Erweiterung des *Faster R-CNN* und kann verschiedene Instanzen einer Klasse in einem Bild von einander trennen. Dazu muss zuerst die Begriffe der Instanzsegmentierung definiert werden.

Einfache Klassifizierung (engl.: classification) ordnet Bilder als Ganzes einer Klasse zu. *Semantische Segmentierung* (engl.: semantic segmentation) beschreibt die Klassifizierung auf Pixelebene. Es wird erkannt zu welcher Klasse eine Menge von Pixeln gehören, aber es wird nicht zwischen einzelnen Objekten unterschieden. *Objekterkennung* (engl.: object detection) entdeckt und lokalisiert unterschiedliche Objekte, indem es eine Bounding Box um jedes erkannte Objekt zieht. Jedoch fehlt hier die pixelgenaue Abgrenzung einzelner

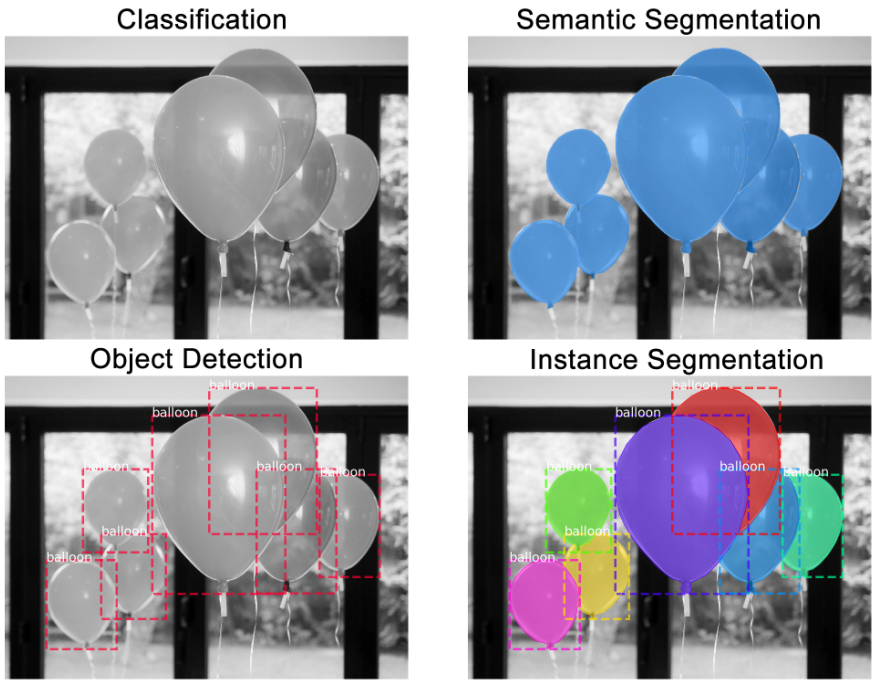

Abbildung 1.4: Unterschied Klassifizierung / semantische Segmentierung / Objekterkennung / Instanzsegmentierung[1]

Objektinstanzen. Instanzsegmentierung (engl.: instance segmentation) kombiniert *Objekterkennung* und *semantische Segmentierung* und ist so in der Lage zwischen einzelnen Objekten zu unterscheiden und ihnen entsprechende Pixel zuzuordnen (s. Abb. 1.4) und ist eine der größten Herausforderungen in der Bildverarbeitung.[11]

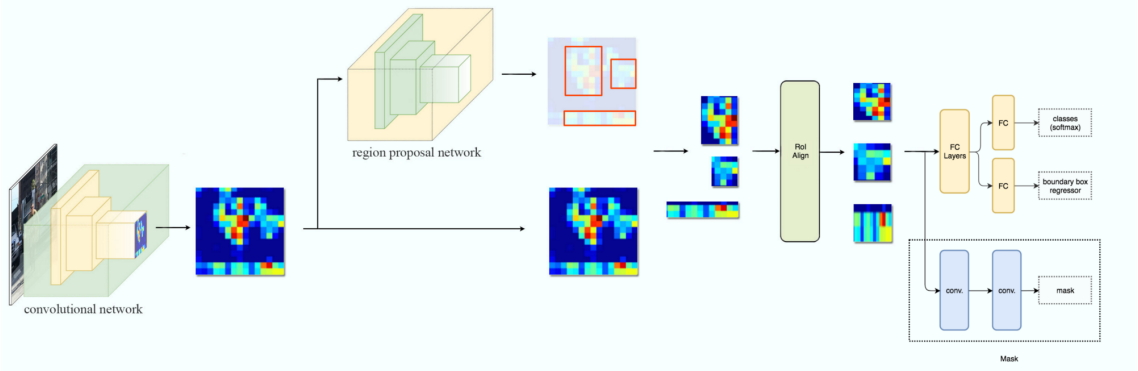

Abbildung 1.5: Mask R-CNN-Architektur[13]

Mask R-CNN ist wie Faster R-CNN in zwei Segmente eingeteilt. In dem ersten Segment, dem *Region Proposal Network* (RPN, dt.: Region vorschlagendes

Netzwerk), werden mehrere Rahmen (Bounding Boxes) innerhalb eines Bildes vorgeschlagen, die interessante Objekte beinhalten könnten.[17] Die vorgeschlagene Regionen, die einzeln von CNNs bewertet werden, ist der Kernansatz von R-CNN. Das RPN wurde identisch von Faster R-CNN für Mask R-CNN übernommen.[11]

Im zweiten Segment werden aus den Regionen *Bounding Boxes* (dt.: Rahmen) und Masken generiert und klassifiziert. Die Rahmen haben verschiedene Größen und können Probleme bei der Klassifizierung verursachen. Daher werden die Rahmen auf eine kleine Feature Map gleicher Größe (z.B. $7 * 7$ px) reduziert. Die Authoren von [11] schlagen eine Methode namens *RoI-Align* vor, bei der Proben aus der Feature Map entnommen werden und eine bilineare Interpolation angewendet wird. In dem bei Faster R-CNN angewandten Verfahren *RoI-Pooling* entstehen durch Quantisierung Informationsverluste und räumliche Abweichungen zwischen Bounding Box und Feature Map, was negative Auswirkungen auf die Maskengenerierung haben kann.[11]

Die oberen vollständig vernetzten Schichten (*FC Layers* in Abb. 1.5) klassifizieren die Regionen und die Bounding Boxes berechnet. Dieser Zweig ist für die Objekterkennung wichtig und noch mit Faster R-CNN gemeinsam.

Gleichzeitig werden in einem parallelen Zweig je Bounding Box k $m * n$ große Masken zur semantischen Segmentierung erzeugt, wobei k die Anzahl der Klassen ist. Anders als in dem ersten Zweig des zweiten Segmentes werden die Masken durch *fully convolutional networks* (FCN, dt.: vollständig faltende Netzwerke) prognostiziert. Diese bestehen nur aus faltenden Schichten, wie sie in Kapitel 1.4.2 beschrieben sind. Eine Maske ist eine räumliche Kodierung eines Objektes und daher ist es wichtig räumliche Informationen beizubehalten. Diese können durch die Pixel-zu-Pixel-Übereinstimmung extrahiert werden, welche sonst durch vollständig vernetzter Schichten verloren gehen. Diese geben einen Vektor ohne räumliche Dimensionen aus.[11]

FCN und wie wird Maske erzeugt?

In [11] wird Mask R-CNN mit den *COCO challenge*-Gewinnern[4] der Jahre 2015 und 2016 verglichen. Der Vergleich zeigt, dass Mask R-CNN in der Chal-

[4]COCO (Common Objects in Context, dt.: Gewöhnliche Objekte im Kontext) enthält einen Datensatz von über 200000 Bildern in über 80 Kategorien. Der Datensatz ist eine oft genutzte Basis, um Objekterkennungstechniken zu evaluieren und zu bewerten.[4]

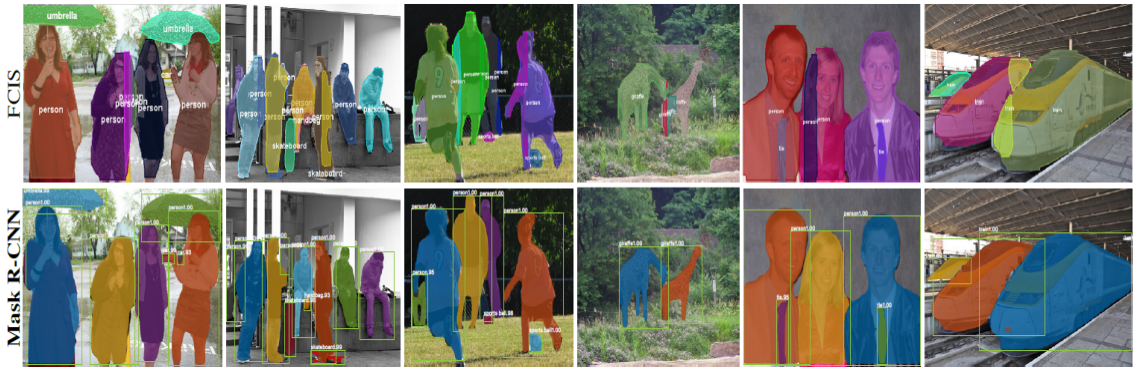

Abbildung 1.6: Bei FCIS entstehen Artefakte, wenn Objekte sich in einem Bild überlappen.[11]

	backbone	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
MNC [10]	ResNet-101-C4	24.6	44.3	24.8	4.7	25.9	43.6
FCIS [26] +OHEM	ResNet-101-C5-dilated	29.2	49.5	-	7.1	31.3	50.0
FCIS+++ [26] +OHEM	ResNet-101-C5-dilated	33.6	54.5	-	-	-	-
Mask R-CNN	ResNet-101-C4	33.1	54.9	34.8	12.1	35.6	51.1
Mask R-CNN	ResNet-101-FPN	35.7	58.0	37.8	15.5	38.1	52.4
Mask R-CNN	ResNeXt-101-FPN	**37.1**	**60.0**	**39.4**	**16.9**	**39.9**	**53.5**

Tabelle 1.2: Instance segmentation *mask* AP auf COCO *test-dev*. MNC und FCIS sind Sieger der COCO 2015 und 2016 Challenge. Mask R-CNN erzielt deutlich bessere Ergebnisse als die komplexere FCIS+++.[11]

lenge bessere Werte erzielt als die Konkurenten (s. Tab. 1. Desweiteren fällt *fully convolutional instance segmentation* (FCIS, dt.: vollständig faltende Instanzsegmentierung) auf, wenn es mit überlappenden Objekten konfrontiert wird. Dort erzeugt es Artefakte, welche durch Mask R-CNN nicht entstehen (s. Abb. 1.6). Durch diese Gegenüberstellungen wird gezeigt, dass Mask R-CNN alle aufgeführten Anforderungen erzielt. Es erkennt Klasseninstanzen auf Pixelebene und weist eine hohe Robustheit auf. Auch die Genauigkeit hebt sich beim direkten Vergleich ab. Aus diesen Gründen wurde Mask R-CNN im Rahmen diese Arbeit ausgewählt.

Kapitel 2

Overfitting

2.1 Begriffserklärung

Genaue Daten über Krankheitsbefälle im Agrarsektor sind rar, da diese in der Regel nicht öffentlich zugänglich sind.[1] Daher musste mit *Overfitting* gerechnet werden. Das künstliche neurale Netzwerk soll daraufhin trainiert werden, dass es möglichst alle Befälle, die untersucht werden, erkennt. Dafür wird es im ersten Schritt mit einem Trainingsdatensatz trainiert. Im folgenden Schritt mit einem kleineren Validierungsdatensatz überprüft, wie gut das Netz trainiert wird. Overfitting tritt auf, wenn das Netz auf die Daten aus dem Trainingsdatensatz mit sehr hoher Erfolgsquote erkennt, jedoch vergleichsweise schlechte Ergebnisse bei der Validierung bzw. bei unbekannten Daten erzielt. Das geschiet, weil sich das Netz auf nicht relevante Datenpunkte konzentriert, die im nur Trainingsdatensatz auftreten, aber nicht die allgemeine Charakteristika der Objekte widerspiegeln.

In Abb. 2.1 ist ein Beispiel wie Overfitting sich auswirken kann. Die linken zwei Bilder sind ein exemplarischer Auszug aus dem Trainingsdatensatz. Einmal eine visuelle Repräsentation der NDVI-Werte der infizierten Agrarfläche und die Binärmaske, welche die infizierte Fläche markiert. Das selbe Bild wurde nach einem erfolgreichen Trainingsdurchlauf der Mask R-CNN-Implementierung übergeben und es hat den erkrankten Bereich nahezu perfekt erkannt. Das vierte Bild zeigt zentriert das selbe Feld. Jedoch ist der Ausschnitt größer, rotiert und die Aufnahme stammt von einem anderen Datum. Der Prognose zur Folge ist die Infizierung auf die benachbarten Felder über-

Erwähnen, dass Mask RCNN Bild und Masken für Training benötigt.

Genaues

[1] Datenschutz kann ein Grund dafür sein.

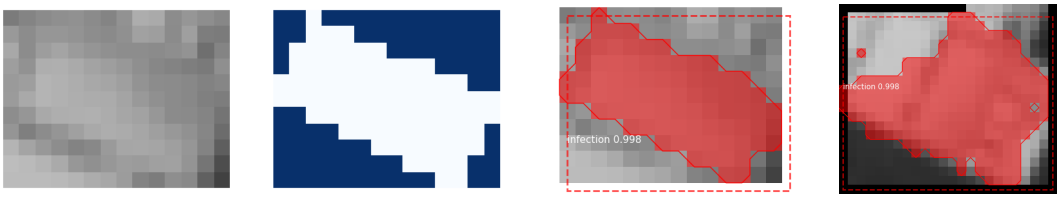

Abbildung 2.1: V.l.n.r. Bild von infizierter Agrarfläche aus Trainingsdatensatz / Binärmaske der inifizierten Region, wird gemeinsam mit dem linken Bild zum Training in das KNN gespeist / Selbiges Bild, Ergebnis nach Trainingsdurchlauf, prognostizierte Ergebnisfläche in rot / Bild der selben Fläche, was nicht aus dem Trainingsdatensatz stammt, prognostizierte Ergebnisfläche in rot

gesprungen, was nicht der Realität entspricht. Overfitting ist ein bekanntes Problem im Bereich des maschinellen Lernens und es existieren multiple Methoden, um dem entgegenzuwirken.

2.2 Data Augmentation

Generell ist ein sehr großer Datensatz (≥ 10000 Elemente) für das Training förderlich. Jedoch ist das nicht immer möglich. In diesem Fall kann der Datensatz künstlich durch *Data Augmentation* vergrößert werden. Zu Data Augmentation zählen geringe Veränderungen der Daten - hier Bildmanipulationen. Solche Operationen können unter anderem

- Rotationen,
- Spieglungen,
- Translationen,
- zufällige Ausschnitte,
- Gauß'sches Rauschen,
- Helligkeitsveränderungen,
- oder Kombinationen davon

beinhalten. Wobei nicht jede Augmentation-Technik für jeden Anwendungsfall sinnvoll ist. Wenn zum Beispiel ein KNN auf Autoerkennung trainiert werden soll, ist es nicht nützlich oder sogar hinterlich die Bilddaten so zu rotieren, dass es von oben nach unten oder umgekehrt zeigt. Desweiteren ist

sinnvoll Data Augmentation einzusetzen, obwohl genügend Daten vorhanden sind. So besteht der Datensatz aus zwei Klassen von Autos, Marke A und Marke B. Die Front der Fahrzeuge der Marke A sind nach rechts ausgerichts, während die Autos der Marke B nach links ausgerichtet sind. Das neuronale Netz wird diesen markanten Unterschied den Marken zuordnen und wird ein links ausgerichtetes Fahrzeug der Marke A als ein Fahrzeug der Marke B klassifizieren.[3]

Für den Anwendungsfall dieser Arbeit sind Rotationen, Spiegelungen, zufällige Ausschnitte und Translationen valide Optionen zur Vergrößerungen des Datensatzes. Wie in dem vorherigen Beispiel wird das Modell auch Merkmale wie Form, Position im Bild oder Ausrichtung der RoI untersuchen, welche in der Realität keine feste Muster haben und nicht vorhersagbar sind. Damit wird nicht nur der Datensatz vergrößert, sondern auch das Modell generalisiert.

Künstliches Rauschen und Helligkeitsveränderungen sind mit hoher Wahrscheinlichkeit nicht geeignet. Die Bilddaten bestehen aus quantifizierten Werten und die Operationen könnten diese Werte verfälschen und damit das Endergebnisse negativ beeinflussen.

2.3 L2 Regularization

Ein kleiner Datensatz führt zu einem komplexen Modell und komplexere Modelle neigen zu Overfitting. *L2 Regularization* (oder auch *Ridge Regression*) vereinfacht das Modell, indem es hohe Gewichte bestraft und niedrige Gewichte bevorzugt.[2] Hohe Gewichte können mit Anomalien korrelieren, die nur im Trainingsdatensatz auftreten und so wird das Netz Schwierigkeiten haben, fremde Daten richtig zu erkennen.

$$loss + \lambda \sum_{j=1}^{p} \beta_j^2 \tag{2.1}$$

Ridge Regression addiert einen zusätzlichen Bestrafungsterm (engl.: *penalty term*) zur Verlustfunktion (engl.: *loss function*), wobei $loss$ die Verlustfunktion, der letzte Term der Bestrafungsterm und $\lambda > 0$ sei. Hier ist darauf zu achten,

[2] Es sei erwähnt, dass es neben L2 Regulization noch *L1 Regulization* (oder auch *Lasso Regression*) existiert. Diese Methode wird eingesetzt, um Underfitting zu verhindern.

dass λ sinnvoll gewählt wird. Wenn es zu groß gewählt wird, wird zu viel Gewicht hinzugefügt und das Modell tendiert zum *Underfitting*[3].[2][16]

2.4 Zusätzliche Methoden

Wenn ein KNN initial trainiert wird, werden die einzelnen Gewichte zufällig gewählt und dann dem Idealwert angenähert. Dieser Prozess kann zeitlich verkürzt werden indem vortrainierte Gewichte eingesetzt werden. Je länger ein Modell trainiert werden muss, desto größer ist die Gefahr des *Overfittings*. Darum wird die Mask R-CNN-Implementierung mit Gewichten initialisiert, die auf dem COCO-Datensatz trainiert wurden.

In einem großen Datensatz beeinflussen einzelne Anomalien das Training nicht. Anders können diese Anomalien einen starken Einfluss in einem kleinen Datensatz haben. Eine hohe Anzahl von trainierbaren Parametern (Gewichten) reagieren darauf empfindlicher und es gilt das Modell durch Parameterminimierung resilienter zu machen. Um die Anzahl an Gewichten, die trainiert werden, zu minimieren, kann das *Backbone*[4] vereinfacht werden. Hier werden *ResNet50* und *ResNet101*, welche jeweils 50 und 101 Schichten besitzen, miteinander verglichen. Hierbei sollte ResNet50 vorteilhafter sein, da es von beiden vorgestellten Architekturen die simplere hat.

Bei einem kleinen Datensatz hilft es die Experimente simpel zu gestalten. Das betrifft auch die Anzahl der Klassen. In dem betroffenen Region wurden zwei verschiedene Krankheiten festgestellt, die als einzelne Klassen deklariert werden können. Um die Datengröße pro Klasse zu erhöhen, werden diese beiden Klassen zu einer Klasse *infection* zusammengefasst.

[3] Underfitting bezeichnet eine schlechte Performanz des neuronalen Netzes auf sämtliche Daten inkl. dem Trainingsdatensatz.

[4] He et al. bezeichnen die Architektur, die für die Merkmalextraktion verantwortlich ist, als Backbone. Das Segment, das Klassifizierung, Bounding-Box-Generierung und Maskenerkennung durchführt, wird als *network head* (oder nur *head*) definiert.[11]

Kapitel 3

Konzept und Implementierung

3.1 Konzept

Das Programmablauf wird in einzelne Schritte unterteilt, auf die in den nächsten Kapiteln näher eingegangen werden.

Zuerst müssen die Daten für das Training bzw. für die Erkennung manuell an-

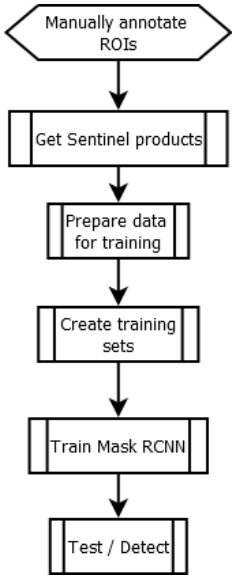

Abbildung 3.1: Gesamtablauf der Anwendung

notiert werden. Diese Metadaten werden dann genutzt, um automatisch Sentinelprodukte[1] mittels einer API, die von der Copernicus zur Verfügung gestellt wird, herunterzuladen. Aus den Produkten werden die relevanten Bänder extrahiert und unter anderem die jeweiligen NDVI-Werte berechnet. Nachdem

[1] Aufnahmenpakete der Sentinel-Plattformen werden als Produkte bezeichnet.

die Produkte für das Training vorbereitet wurden, werden die Daten in ein Trainings- und in ein Validierungsdatensatz aufgeteilt. Der folgende Trainingsprozess basiert auf diesen Datensätzen. Sobald das Training abgeschlossen ist, kann die Performanz des Modells getestet werden.

3.2 Annotation

Zu Beginn werden die Regionen, die entweder für das Training benutzt oder überprüft werden, manuell erfasst. Vorrausgesetzte Informationen sind

- Geografische Koordinaten,
- Zeitraum des Befalls und
- Bezeichnung der Infektion.

Als Format dieser Informationen dient *GeoJSON*[2]. GeoJSON enthält nicht nur geografische Daten, sondern ist auch um benutzerdefinierte Eigenschaften (`properties`) erweiterbar. Die Annotationen sind also GeoJSON-Features, die ein geografisches Polygon und Metadaten enthalten.

```
{
  "type": "Feature",
  "properties": {
    "disease": 1,
    "from": "2018-07-12T13:00:00Z-7DAYS",
    "to": "2018-07-12T13:00:00Z+7DAYS"
  },
  "geometry": {
    "type": "Polygon",
    "coordinates": [[[11.171988617177981,44.574291380353003],
       [11.1726616444942,44.574017992242283],
       [11.17338129910439,44.575068359984279],
       [11.17273129334275,44.575299863118993],
       [11.171988617177981,44.574291380353003]]]
  }
}
```

[2]GeoJSON ist eine Erweiterung des JSON-Format und beschreibt geografische Daten und Geometrien. GeoJSON wird durch den RFC7946-Standard definiert.

Listing 3.1: Annotation

`properties.disease` enthält die eindeutige, nummerische Repräsentation der Klasse bzw. Krankheit, die in dieser Region enthalten ist. Die Zuordnung der nummerischen Werte und des textuellen Bezeichners werden in einer separaten JSON als Schlüssel-Wert-Paare konfiguiert, wobei der Schlüssel nummerisch und der Wert textuell ist. Hier ist, darauf zu achten, dass der Schlüssel ≥ 1 ist, da 0 der implizite Schlüssel der Mask R-CNN-Implementierung für den Hintergrund ist. Diese Eigenschaft ist nur für das Training von Relevanz.

`properties.from` und `properties.to` sind jeweils Start- und Endzeitpunkt, in dem nach verfügbaren Sentinelprodukten gesucht werden soll. Das Format der jeweiligen Eigenschaften kann eine der folgenden Formen haben[3]:

- `yyyyMMdd`
- `yyyy-MM-ddThh:mm:ss.SSSZ` (ISO-8601)
- `yyyy-MM-ddThh:mm:ssZ`
- `NOW`
- `NOW-<n>DAY(S)` (oder `HOUR(S)`, `MONTH(S)`, usw.)
- `NOW+<n>DAY(S)`
- `yyyy-MM-ddThh:mm:ssZ-<n>DAY(S)`
- `NOW/DAY` (oder `HOUR`, `MONTH` usw.) - Der Wert wird auf den jeweiligen Typ (z.B. auf den Tag) gerundet.

Es ist angebracht einen Zeitraum von mehreren Tagen bzw. Wochen zu wählen, da die Sentinel-2-Satelliten keine täglichen Daten liefern und weil eine Infektion typischerweise über einen längeren Zeitraum vorherrscht. Die Zeitspanne ist von der Krankheit abhängig. Hier wurden eine Woche vor und nach dem Aufnahmezeitpunkt genutzt, um nach Produkten zu suchen.

3.3 Suche nach Sentinelprodukte

Der *Copernicus Open Access Hub*[4] ermöglicht freien und offenen Zugriff auf Sentinel-Produkte. Die Daten sind sowohl über eine grafische Oberfläche als auch über eine REST-API verfübar. Vorrausgesetzung für beide Optionen ist ein Account, der über die grafische Oberfläche erstellt werden kann.

Die Nutzung der Schnittstelle erfolgt über die Python-Bibliothek `sentinelsat`[5]. Bei einer Anfrage müssen die GeoJSON-Dateien in WKT[6] umgewandelt werden, was von der Bibliothek übernommen werden kann. Die WKT-Geometrie wird als *footprint* (dt.: Fußabdruck) bezeichnet. Desweiteren wird der Plattformname statisch als 'Sentinel-2' definiert, damit keine Produkte von den anderen Sentinelplattformen zurückgegeben werden. Der Suchzeitraum wird aus der jeweiligen GeoJSON-Datei übernommen. Sollten für die Suchanfragen keine Produkte existieren, wird die Anwendung beendet, da es keine Basis gibt, auf der das Netzwerk trainiert werden kann. Eventuell muss bei so einem Fall der Zeitraum erweitert und Prozess wiederholt werden. Bei vorhandenen Produkten lädt das Skript diese herunter. Die Produkte werden in einem komprimierten Format geliefert und enthalten neben zusätzlichen Informationen, Banddaten in separaten Dateien im JPEG2000-Format[7].

Für jedes Produkt, das zur aktuellen Annotation gehört, wird ein neuer Eintrag zu einer *FeatureCollection* hinzugefügt. Zusätzlich wird eine *UUID* (Universally Unique Identifier) generiert und zusammen mit dem Dateinamen der Produktes in den Metadaten gespeichert. Für die spätere Entwicklung sind die Annotationen leichter zu finden und bearbeitbar. Außerdem bleiben dadurch die Originaldaten unberührt. Dieser Schritt wird für jede vorhandene Annotationsdatei wiederholt. Anschließend werden die Bilddateien für B4 und B8 aus dem Produkt extrahiert.

[3] Die Formate basieren auf der `sentinelsat`-Version 0.12.2.

[4] `https://scihub.copernicus.eu/`

[5] https://sentinelsat.readthedocs.io/en/stable

[6] WKT (Well-known text) ist eine Markup-Sprache zur Repräsentation von geometrischen Objekten auf Karten und räumlichen Referenzsystemen.

[7] JPEG 2000 genau wie GeoTIFF ist ein Bildformat in dem auch Metadaten abgelegt werden können. So sind Pixel geografischen Koordinaten zuordbar.

3.4 Aufbereitung der Sentineldaten

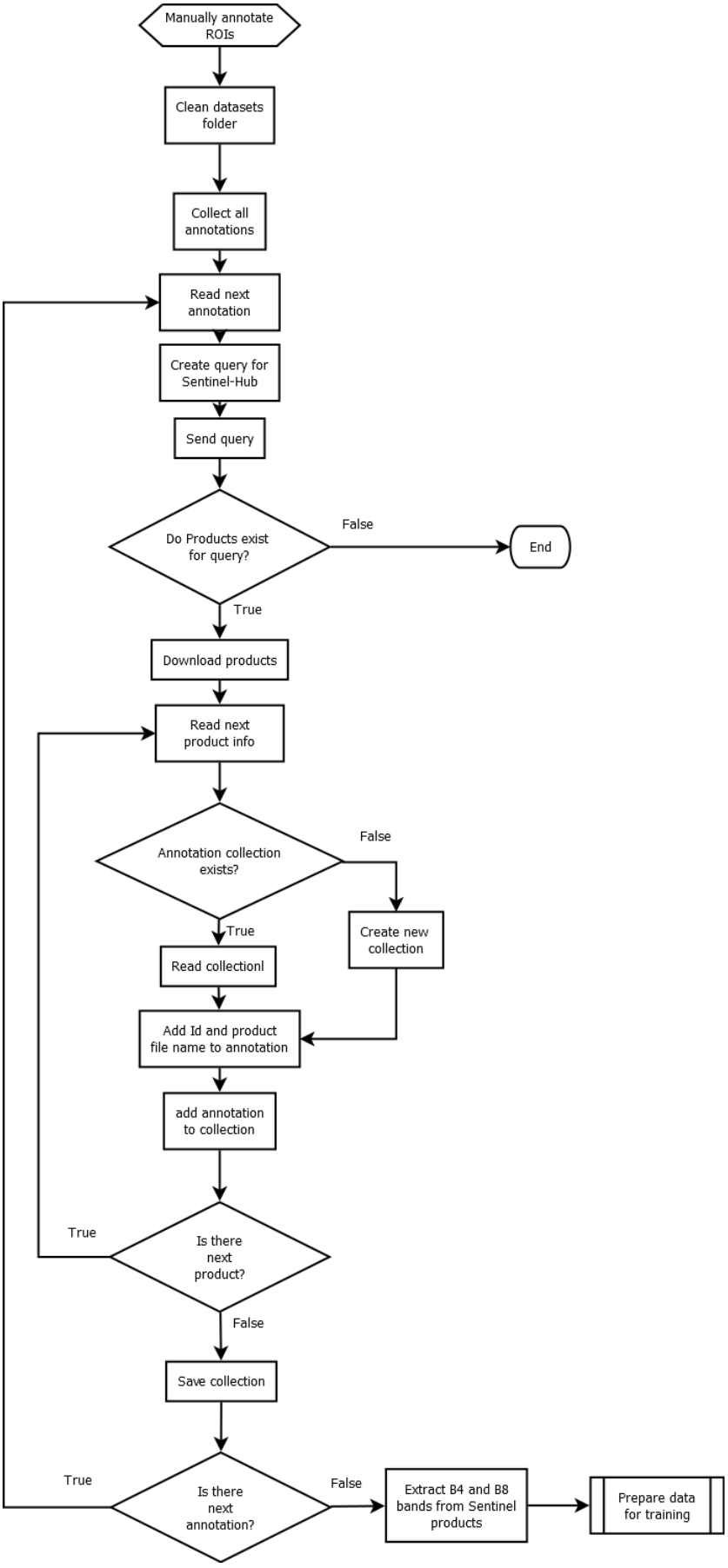

Abbildung 3.2: Ablaufdiagramm der Beschaffung der Sentinelprodukte

Literaturverzeichnis

[1] W. Abdulla. Splash of color: Instance segmentation with mask r-cnn and tensorflow. https://engineering.matterport.com/splash-of-color-instance-segmentation-with-mask-r-cnn-and-tensorflow-7c761 2018. [Zuletzt besucht: 22.12.2018].

[2] Anuja Nagpal. L1 and l2 regularization methods. https://towardsdatascience.com/l1-and-l2-regularization-methods-ce25e7fc831c, 2017. [Zuletzt besucht: 09.01.2019].

[3] Bharath Raj. Data augmentation | how to use deep learning when you have limited data - part 2. https://medium.com/nanonets/how-to-use-deep-learning-when-you-have-limited-data-part-2-data-augmentati 2018. [Zuletzt besucht: 09.01.2019].

[4] C. Consortium. Coco 2018 object detection task. http://cocodataset.org/#detection-2018, 2018. [Zuletzt besucht: 06.01.2019].

[5] Copernicus. Copernicus in brief. https://www.copernicus.eu/en/about-copernicus/copernicus-brief. [Zuletzt besucht: 15.12.2018].

[6] N. Corporation. Convolutional neural network (cnn). https://developer.nvidia.com/discover/convolutional-neural-network, 2019. [Zuletzt besucht: 04.01.2019].

[7] ESA. Level-2a algorithm overview. https://sentinel.esa.int/web/sentinel/technical-guides/sentinel-2-msi/level-2a/algorithm, 2018. [Zuletzt besucht: 20.12.2018].

[8] ESA. Radiometric resolutions. https://earth.esa.int/web/sentinel/user-guides/sentinel-2-msi/resolutions/radiometric, 2018. [Zuletzt besucht: 20.12.2018].

[9] ESA. Resolutions. https://earth.esa.int/web/sentinel/user-guides/sentinel-2-msi/resolutions, 2018. [Zuletzt besucht: 20.12.2018].

[10] A. A. Gitelson, Y. Gritz, and M. N. Merzlyak. Relationships between leaf chlorophyll content and spectral reflectance and algorithms for non-destructive chlorophyll assessment in higher plant leaves. *Journal of Plant Physiology*, 160(3):271 – 282, 2003.

[11] K. He, G. Gkioxari, P. Dollár, and R. B. Girshick. Mask R-CNN. *CoRR*, abs/1703.06870, 2017.

[12] G. A. F. Hendry, J. D. HOUGHTON, and S. B. BROWN. The degradation of chlorophyll — a biological enigma. *New Phytologist*, 107(2):255–302, 1987.

[13] J. Hui. Image segmentation with mask r-cnn. https://medium.com/@jonathan_hui/image-segmentation-with-mask-r-cnn-ebe6d793272, 2018. [Zuletzt besucht: 06.01.2019].

[14] C. Mattupalli, C. A. Moffet, K. N. Shah, and C. A. Young. Supervised classification of rgb aerial imagery to evaluate the impact of a root rot disease. *Remote Sensing*, 10(6), 2018.

[15] NASA. Measuring vegetation (ndvi & evi). https://earthobservatory.nasa.gov/features/MeasuringVegetation/measuring_vegetation_2.php, 2000. [Zuletzt besucht: 13.12.2018].

[16] Prashant Gupta. Regularization in machine learning. https://towardsdatascience.com/regularization-in-machine-learning-76441ddcf99a, 2017. [Zuletzt besucht: 09.01.2019].

[17] S. Ren, K. He, R. B. Girshick, and J. Sun. Faster R-CNN: towards real-time object detection with region proposal networks. *CoRR*, abs/1506.01497, 2015.

[18] U.S. Department of the Interior. What are the band designations for the landsat satellites? https://sentinel.esa.int/web/sentinel/technical-guides/sentinel-2-msi/level-2a/algorithm, n.d. [Zuletzt besucht: 24.12.2018].

[19] J. Verrelst, J. Muñoz, L. Alonso, J. Delegido, J. P. Rivera, G. Camps-Valls, and J. Moreno. Machine learning regression algorithms for biophysical parameter retrieval: Opportunities for sentinel-2 and -3. *Remote Sensing of Environment*, 118:127 – 139, 2012.