Patrick Merita

Das Potenzial von künstlicher Intelligenz am Beispiel der Ableitung des Landbedeckungsmodells (LBM-DE) für das Bundesland Schleswig-Holstein

Masterarbeit

Hochschule Mainz
Fachbereich Technik
Fachrichtung Geoinformatik und Vermessung

1.	Gutachter	Prof. Dr. Pascal Neis	HS Mainz
2.	Gutachter	Dr. Michael Wurm	DLR
3.	Gutachter	Dr. Michael Hovenbitzer	BKG

Standnummer: 275

Mainz
Februar 2024

BKG Wir geben Orientierung. Bundesamt für Kartographie und Geodäsie

DLR Deutsches Zentrum für Luft- und Raumfahrt

Zusammenfassung

Durch die gestiegene Zugänglichkeit der Daten verschiedener Erdbeobachtungsdaten sind methodische Ansätze zur deren automatisierten Analyse essenziell. Das Potenzial der künstlichen Intelligenz ist mittlerweile fester Bestandteil in der Arbeit mit Erdbeobachtungsdaten. Durch die großen Entwicklungen im Hardwarebereich und die Möglichkeit zur Prozess Parallelisierung rücken Deep Learning Verfahren zunehmend in den Fokus der Wissenschaft. Hierbei hat sich die semantische Segmentierung von Landbedeckung mittels Encoder-Decoder Ansätzen als besonders leistungsfähig etabliert. Mit den Landbedeckungsdaten des Landbedeckungsmodells Deutschland (LBM-DE) steht dieser Arbeit ein hochwertig gelabelter Datensatz für das Bundesland Schleswig-Holstein für das Jahr 2021 zur Verfügung. Auf der Basis dieser Daten wurde evaluiert, inwieweit ein vortrainiertes Modell auf RapidEye Daten zur Klassifizierung von PlanetScope Daten genutzt werden kann. Die Evaluation erfolgt anhand drei verschiedener Szenarien mit einer unterschiedlichen Anzahl an Landbedeckungsklassen. Mit klassenspezifischen F1-Scores von bis zu 0,98 zeigt sich das große Potential für die Ableitung von Landbedeckung mit Ansätzen der künstlichen Intelligenz. Speziell für den geplanten Einsatz unterschiedlicher Erdbeobachtungsmissionen zur Ableitung der Landbedeckung innerhalb des BKGs sind diese Ergebnisse vielversprechend.

Abstract

Due to an increase in the accessibility of earth observation data from various earth observation missions, methodological approaches for their automated analysis are essential. The potential of artificial intelligence has become a fundamental part in the work with Earth observation data. Because of major developments in the hardware sector and the possibility to parallelize processes, deep learning methods are increasingly moving into the focus of science. Semantic segmentation of land cover using encoder-decoder approaches has established as particularly powerful. With the land cover information of the Land Cover Model Germany (LBM-DE), a high-quality labeled data set for the federal state of Schleswig-Holstein for the year 2021 is available for this work. Based on this data, it was evaluated to what extent a pre-trained model on RapidEye data can be used to classify PlanetScope data. The evaluation is based on three different scenarios with a different number of land cover classes. With class-specific F1-scores of up to 0,98, the great potential for deriving land cover with artificial intelligence approaches is evident. These results are particularly promising for the planned use of different earth observation missions to derive land cover within the BKG.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

BB	**Brandenburg**
BKG	**Bundesamt für Kartographie und Geodäsie**
CLC	**Corine Land Cover**
CNN	**Convolutional Neural Networks**
CPU	**Central Processing Unit**
DBMS	**Datenbankmanagementsystem**
DFN	**Deep Feedforward Networks**
DL	**Deep Learning**
DLR	**Deutsches Zentrum für Luft- und Raumfahrt**
DN	**Digital Numbers**
DWD	**Deutscher Wetterdienst**
FC-DenseNet	**Fully Convolutional Densely Connected Convolutional Networks**
FCN	**Fully Convolutional Networks**
FLOPS	**Floating Point Operations per Second**
FN	**False Negative**
FP	**False Positive**
GD	**Gradientenabstieg (engl. Gradient Descent)**
GPU	**Graphics Processing Unit**
HE	**Hessen**
ISPRS	**International Society for Photogrammetry and Remote Sensing**
IoU	**Intersection over Union**
KI	**Künstliche Intelligenz**
KNN	**Künstliches Neuronales Netz**
LBM-DE	**Digitales Landbedeckungsmodell Deutschland**
ML	**Machine Learning**
MLP	**Multilayer Perceptron**
MV	**Mecklenburg-Vorpommern**
PCA	**Principle Component Analysis**
Pre	**Präzision (engl. Precision)**
Rec	**Recall**
PX	**Pixel**
RefineNet	**Refinement Network**
ReLU	**Recitified Linear Unit**
ResNet	**Residual Networks**
RNN	**Recurrent Neural Network**
SegNet	**Deep Fully Convolutional Network zur Semantic Segmentation**
SGD	**Stochas. Gradientenabstieg (eng. Stochastic Gradient Descent)**
SH	**Schleswig-Holstein**
Tanh	**Hyperbolischer Tangens als Aktivierungsfunktion**
TN	**True Negative**
TOA	**Top-of-Atmosphere**
TP	**True Positive**
Train_Loss	**Trainingsverlust**
Val_Loss	**Validierungsverlust**

Formelverzeichnis

1 Einleitung

In der jüngsten Vergangenheit hat die Rolle von künstlicher Intelligenz (KI) stark an Bedeutung gewonnen. Insbesondere die Entwicklung von ChatGPT vom amerikanischen Unternehmen OpenAI hat den Begriff KI in der Gesellschaft noch präsenter gemacht als zuvor. Dabei gehen erste Definitionen bereits in die 1960er Jahre zurück. Nach John McCathy (Junior-Professor am Dartmouth College in New Hampshire) geht es „um die Entwicklung von Maschinen, die sich verhalten, als verfügten sie über menschliche Intelligenz“ (KAPLAN 2017). Hierbei stellt sich die Frage, wie sich die aktuelle Popularität der KI erklären lässt. Zwei fundamentale Aspekte haben die Entwicklung von KI-Algorithmen revolutioniert: Datenverfügbarkeit und Rechenleistung. Erste KI-Anwendungen sind mit heutigen Ansätzen nicht mehr vergleichbar. So entwickelte man sich von ersten logischen Repräsentationssystemen über den großen Bereich des maschinellen Lernens bis hin zum tiefen Lernen (dem sogenannten *Deep Learning*) (GAßNER 2019). Deep Learning (DL) hat sich in vielen wissenschaftlichen Bereichen als zukunftsfähige, leistungsstarke Methode etabliert und ist gerade in der Parallelisierung einfacher Aufgaben konkurrenzlos (HOESER UND KUENZER, 2020). Speziell bei der Erkennung von Bild und Sprache sind DL-Modelle klassischem maschinellem Lernen überlegen (CIRESAN ET AL. 2012; DAHL ET AL. 2012). Jedoch sei angemerkt, dass die Forschungsfrage sowie die auszuführenden Aufgaben die Wahl des KI-Algorithmus bestimmen und Deep Learning nicht immer die beste Wahl ist. Bezogen auf die Erdbeobachtung haben sich künstliche neuronale Netze (CNNs), als Untergruppe des DLs, bereits als anpassungsfähige Methode für die neuen Herausforderungen bewährt. Durch die steigende Anzahl von Erdbeobachtungsprogrammen, die damit verbundene Datenverfügbarkeit und die Entwicklung sogenannter „Graphical Processing Units“ (GPUs) kann das Potential der CNNs immer besser in der Fernerkundung eingesetzt werden. Allein im letzten Jahrzehnt ist ein starker Anstieg an Publikationen mit DL-Bezug zu verzeichnen (HOESER ET AL. 2020). Dies wird unteranderem durch die Verfügbarkeit von räumlich hochaufgelösten optischen und multispektralen Daten begünstigt. So ergaben sich nach HOESER ET AL. (2020) 429 Publikationen mit DL für die Erdbeobachtung seit 2012. Betrachtet wurden explizit Publikationen, die sich auf die Bildsegmentierung und die Objektdetektion beziehen. Eingeteilt in neun Kategorien werden die zahlreichen Anwendungsbereiche verdeutlicht. Die drei Hauptkategorien beziehen sich auf den Verkehrssektor, die Erkennung von Siedlungsstrukturen

sowie die Klassifikation von Landbedeckung und -nutzung. Klassische Beispiele für DL im Verkehrssektor sind die Detektion von Autos (TANG ET AL. 2017), Flugzeugen (CAI ET AL. 2018) und Schiffen (YOU ET AL. 2019) oder die Segmentation von ganzen Straßennetzen (WIE ET AL. 2017). Generell ist zu sagen, dass sich das Verhältnis zwischen den Publikationen zur Bildsegmentierung und Objektdetektion zugunsten der Bildsegmentierung verschiebt (HOESER ET AL. 2020). Dies zeigt sich auch in den Veröffentlichungen im Bereich Siedlungsstrukturen, wo neben Gebäudedetektionen (YANG ET AL. 2018) auch Veränderungsanalysen zur Ausbreitung bspw. bebauter Bereiche relevanter werden (DAUDT ET AL. 2018). Hierbei lassen sich Rückschlüsse auf die verwendeten DL-Architekturen und deren Verfügbarkeit finden. Im Jahr 2015 nahm das ResNet von He et al. (2015) an der ImageNet-Challenge teil und überzeugte mit dem besten Klassifizierungsergebnis. ResNet erhielt seinen Namen durch die „Residual Units" aus denen das Netz aufgebaut ist. Diese steigern die Konnektivität im Netz, erleichtern die Backpropagation und ermöglichen dem Klassifizierungslayer eine direkte Verbindung zu den oberen Schichten im Netz (HE ET AL. 2015). Nutzt man das ResNet als „convolutional backbone" zur Extraktion vorhandener Features, sollte man diesen unter Berücksichtigung der Inputdaten gestalten. Hier spielen die Tiefe, die Anzahl der zu extrahierenden Klassen sowie deren Repräsentation und Rechenlast eine entscheidende Rolle. Vor dem Hintergrund, dass der Ursprung des ResNets im Bereich des Maschinellen Sehens liegt, wo eine enorme Menge an unterschiedlichen Klassen erkannt werden müssen, ist dies bei klassischen Landbedeckungsklassifikationen oft nicht der Fall. Daher liefern die tiefsten ResNets wie bspw. das ResNet-152 (bestehend aus 152 Layern) oft schlechtere Genauigkeiten als flachere Netze, aufgrund der geringeren Anzahl an zu erkennenden Klassen in Fernerkundungsdaten (HOESER ET AL. 2020). Um Overfitting zu vermeiden, tendiert man eher zu flacheren Netzen. Das Modeldesign muss hier die gewünschten Aufgaben inkl. Daten angepasst werden. Nach der Wahl des geeigneten Backbones kann dieser in einem Encoder-Decoder Modell verwendet werden. Das Backbone agiert hier als Encoder und extrahiert sogenannte „Feature Maps" aus den Inputdaten. Encoder-Decoder Systeme gehören neben den „Naiven Decodern" zu den Architekturen der FCNs („Fully Convolutional Networks") aus dem Jahr 2014. Aufgrund der starken Vernetzung dieser Systeme helfen speziell im „Upsampling" die Informationen des Encoders in Bezug auf die Auflösung der Eingangsdaten. Hierdurch werden feine räumliche Strukturen besser segmentiert als in „Naive Decoder Modellen". Diese Eigenschaft kommt der Beschaffenheit von

Fernerkundungsdaten entgegen und bestätigt die Relevanz der Encoder-Decoder Systeme für die Bildsegmentierung.

Betrachtet man nun die Einführung beschriebener Architekturen aus den Jahren 2014 (FCNs) und 2015 (ResNets) und den Anstieg von Publikationen im Bereich Bildsegmentierung seit 2015, lassen sich Parallelen erkennen. Dies wird ebenfalls durch die dritte Hauptgruppe der betrachteten Publikationen durch Hoeser et al. (2020) bestätigt. Seit 2014 gibt es eine Zunahme in Publikationen zur Landbedeckungsklassifikation unter Verwendung von DL-Algorithmen. Hier gibt es bereits Ansätze, die sich mit Einsatz von DL-Modellen zur Klassifikation vieler lokaler Klassen mit hoher räumlicher Auflösung auseinandersetzen (HENRY ET AL. 2019; ZHANG ET AL. 2019). Weiterhin lässt sich beobachten, dass bei größeren Untersuchungsgebieten, aufgrund verringerter räumlicher Auflösung der Eingangsdaten, oftmals aggregierte Klassen untersucht werden (HOESER ET AL. 2020). Dies zeigt wie die räumliche Auflösung und die Größe des Untersuchungsgebiets die Entwicklung von DL-Algorithmen beeinflussen können.

Im Folgenden werden neben dem Hintergrund dieser Ausarbeitung, die Ziele der Arbeit und die Forschungsfragen definiert. Speziell geht es um die Ableitung des digitalen Landbedeckungsmodells (LBM-DE) für Schleswig-Holstein auf Basis von Planet Labs Daten.

1.1 Hintergrund

Die folgende Arbeit wird im Rahmen des BKG-Projekts „KI-basierte Analyse in der Fernerkundung“ erstellt. Bereits 2018 wurde in der Nationalen KI-Strategie des Bundes eine Neuausrichtung der Verwaltung festgelegt um „eine Vorreiterrolle beim Einsatz von KI innerhalb der Verwaltung einzunehmen. Diese soll zur „Verbesserung der Effizienz, Qualität und Sicherheit von Verwaltungsdienstleistungen beitragen“ (NATIONALE KI STRATEGIE 2018:32). Zu diesem Zwecke wurde 2021 das Zentrum für Luft- und Raumfahrt (DLR) mit der Entwicklung von KI-Algorithmen zur Auswertung großer Geodatenmengen betraut. Innerhalb eines Unterprojekt namens „DatKI4BKG“ befasst sich das DLR mit der Entwicklung dieser Algorithmen und der Bereitstellung eines Datenbankmanagementsystems (DBMS) zur Verwaltung dieser großen Datenmengen. Solche Maßnahmen ermöglichen dem BKG, „durch den Einsatz von KI, Fernerkundungsinformationen für die Bedarfe der Bundesverwaltung aufzubereiten und auszuwerten“ (FORTSCHREIBUNG DER NATIONALEN KI- STRATEGIE 2020:32).

Der Fokus der Entwicklung der Algorithmen wurde in DatiKI4BKG auf das digitale Landbedeckungsmodell (LBM-DE) gelegt (Kapitel 3.3). Das LBM-DE eignet sich hier besonders, da es in einem Turnus von drei Jahren für die gesamte Bundesfläche aktualisiert und im Vektorformat abgeleitet wird. Dadurch stehen für die Jahre 2015, 2018 und 2021 Referenzdatensätze zur Verfügung, um die Ergebnisse des DL-Verfahrens zu evaluieren.

Für die Entwicklung von KI-Algorithmen, speziell dem DL und den CNNs, wird eine große Anzahl an Eingangsdaten benötigt. Im Rahmen von DatKI4BKG wurden dem DLR seitens BKG RapidEye Daten der Jahre 2015 und 2018 für das gesamte Bundesgebiet zur Verfügung gestellt (Kapitel 3.1). Aufgrund fehlender Verfügbarkeit für das Jahr 2021 wurden hier SPOT-Daten (**S**atellite **P**our l'**O**bservation de la **T**erre) gewählt. Bei SPOT handelt es sich um ein Satellitensystem der französischen Weltraumagentur CNES (**C**entre **N**ational d'**E**tudes **S**patiales). Das Modell wurde initial auf RapidEye Daten des Jahres 2015 trainiert und getestet wurde. Daraufhin wurde das Modell auf die Daten von 2018 angewandt und angepasst. Aktuell befasst sich das DLR mit der Anwendung des Modells auf die SPOT-Daten von 2021, was unteranderem aufgrund des Sensorwechsels und Unterschieden in der Struktur und Qualität der Daten problematisch ist. Durch Anwendung unterschiedlicher Eingangsdaten soll weiterhin evaluiert werden, wie gut das Modell generalisiert und welchen Einfluss vorhandene Variationen der Eingangsdaten auf das Klassifizierungsergebnis haben. Ursprünglich war seitens des DLRs noch die Anwendung auf PlanetScope Daten für das Jahr 2021 geplant, was nun der Fokus dieser Arbeit sein wird.

1.2 Zielsetzung

Die folgende Arbeit befasst mit der Ableitung des LBM-DE mittels Deep Learning für das Bundesland Schleswig-Holstein auf Basis von PlanetScope Daten. Hierfür wird das vom DLR entwickelte Modell verwendet, welches bereits auf RapidEye Daten (Kapitel 3.1) trainiert und getestet wurde. Es handelt sich um ein Encoder-Decoder System (Kapitel 2.3.3) und eignet sich, wie die Literaturrecherche zeigte, für die Lösung von Bildsegmentierungsaufgaben (z.B. WURM ET AL. 2019, VOELSEN ET AL. 2022). Die Wahl des Untersuchungsgebiets fiel auf Schleswig-Holstein, da zu Beginn der Ausarbeitung lediglich das LBM-DE für diesen Bereich Deutschlands für das Jahr 2021 fertig bearbeitet, und

somit als Referenzdatensatz nutzbar war. Folgende Forschungsfragen ergeben sich für die Arbeit:

1. Welche Gesamt- und klassenspezifischen Genauigkeiten erreicht das KI-Verfahren bei der Klassifizierung von PlanetScope Daten?
2. Wie müssen die Daten vorverarbeitet werden? Und welche Hyperparameter eignen sich für den Trainingsprozess?
3. Welche Bedeutung haben die Klassifizierungsergebnisse und wie sind diese Ergebnisse in Bezug auf die Aktualisierung des LBMs zu interpretieren?

Durch den tiefen Einstieg in das Modell sollen Erfahrungswerte im Umgang mit KI-Algorithmen, speziell der Bildsegmentierung erlangt werden. Um das Modell in Zukunft produktiv im BKG nutzen zu können, sind vertiefte Einblicke in die Struktur und Evaluierung der Ergebnisse unabdinglich. Spezieller Fokus liegt auf dem Erlangen von Erfahrungswerten im Rahmen des Sensorübertrags von RapidEye zu PlanetScope. Dies ist entscheidend für zukünftige Überträge und der Anwendung des Verfahrens auf weitere Sensoren.

1.3 Aufbau der Arbeit

Die Arbeit beginnt mit einer Einleitung in die Thematik. Hier wird ein Einstieg in den Bereich der künstlichen Intelligenz gegeben und dessen zunehmende Bedeutung für die Analyse von Erdbeobachtungsdaten beschrieben (1.1). Daraufhin werden die Ziele sowie mit der Arbeit verbundene Leitfragen skizziert (1.2). Das erste Kapitel wird mit der Gliederung der Arbeit abgeschlossen (1.3), bevor es zu den Grundlagen aus dem Bereich der künstlichen Intelligenz übergeht. Nach einem generellen Einstieg in die künstliche Intelligenz (2.1) und der Rolle von Machine Learning (2.2) wird detaillierter auf Deep Learning und den damit verbundenen notwendigen Grundlagen (2.3) eingegangen. Kapitel 2.3 untergliedert sich in Deep Feedforward Networks (2.3.1), Convolutional Neural Networks (2.3.2) und die Semantische Segmentierung (2.3.3). Weiterhin werden ResNets (2.3.4) und FPNs (2.3.5) als in dieser Arbeit verwendete Netzarchitekturen dargestellt. Kapitel 2 wird mit der Auflistung gängiger Deep Learning Frameworks (2.4) sowie Deep Learning in der Erdbeobachtung (2.5) abgeschlossen.

Nach Klärung der Grundlagen erfolgt die Erläuterung der verwendeten Eingangsdaten und die damit verbundene Methodik. Hier werden die Eingangsdaten des initialen Trainings (RapidEye (3.1)) sowie die Basis der Auswertung dieser Arbeit (PlanetScope (3.2)) besprochen. Als Referenzdatensatz ist ebenfalls die Diskussion des LBM-DE und dessen klassenspezifischer Zusammensetzung (3.3) notwendig. Weiterhin wird auf das verwendete Modell (3.4) und die Vorbereitung der Daten (3.5) eingegangen. Abgerundet wird Kapitel 3 mit dem Trainingsprozess und Hyperparameteroptimierung (3.6) und der verwendeten Software (3.7).

Kapitel 4 befasst sich mit der Semantischen Segmentierung als Hauptbestandteil dieser Arbeit. Ist das Training abgeschlossen, muss das Modell evaluiert werden. Hierfür werden die Klassifizierungsergebnisse visuell ausgewertet und diskutiert (4.1). Im Anschluss erfolgt ein Abgleich dieser visuellen Interpretation mit gängigen Metriken zur Bewertung der Modellgüte (4.2). Abschließend erfolgt eine Zusammenfassung der Erkenntnisse sowie ein Ausblick auf zukünftige Untersuchungen (5).

2 Grundlagen

Durch die rasante Entwicklung von KI-Algorithmen und deren Einzug in unzähligen Anwendungsfeldern kam nahezu jeder schon in Kontakt mit Künstlicher Intelligenz. Fragt man den klassischen Anwender jedoch nach einer einheitlichen Definition, ist dies meist nicht trivial. Aus diesem Grund wird in diesem Kapitel die Entwicklung der künstlichen Intelligenz sowie deren Ursprung dargestellt. In diesem Zusammenhang werden deren Untergruppen des „Machine Learnings“ sowie des „Deep Learnings“ durchleuchtet und dargestellt, wie diese voneinander abzugrenzen sind. Abgeschlossen wird das Kapitel mit einem detaillierten Überblick über gängige Deep Learning Frameworks.

2.1 Künstliche Intelligenz

In der heutigen Zeit ist „Künstliche Intelligenz“ (KI) in unzähligen Anwendungsfeldern angekommen. Von autonom fahrenden Autos bis hin zu klassischen Chatbots, die den Nutzer bei seinem nächsten Onlineeinkauf unterstützen sollen, ist alles vorhanden. Trotz dieser umfassenden Präsenz ist die Definition von KI nicht einfach. Nach ROSCHER UND DREES (2022) unterscheidet man zwischen schwacher und starker KI. Schwache KI-Systeme befassen sich im Wesentlichen mit der Datenanalyse und deren Interpretation, wohingegen starke KI dem Menschen überlegen sein soll – die Vorstellung einer sog. Superintelligenz (ROSCHER UND DREES, 2022). Heutige, sowie in dieser Arbeit dargestellte KI ist klar der ersten Kategorie zuzuordnen. Gerade im Hinblick der schnellen und dynamischen Entwicklung im Feld der KI lässt sich eine einheitliche Definition nicht festlegen (MONETT ET AL. 2020). Erste Definitionen gehen in die 1960er zurück. Dort beschrieb John McCarthy KI als Maschinen beschrieb, die in der Lage sind, menschliche Intelligenz zu imitieren (KAPLAN 2017). Um den abstrakten Begriff der KI bzw. der Intelligenz greifbarer zu machen, werden in der Literatur häufig vier Kategorien verwendet. Nach RUSSELL UND NORVIG (2015) soll KI menschlich und rational denken und handeln. Die Stärke des Computers lag lange bei der Abarbeitung komplexer Aufgaben, die auf klarem definiertem Expertenwissen bestand. Ein bekanntes Beispiel ist die Entwicklung erster Schachcomputer, die in der Lage waren, die weltbesten Schachspieler zu besiegen. Trotz der Überlegenheit in der Bearbeitung abstrakter, formaler Vorgaben waren Computer lange nicht in der Lage, für den Menschen intuitive Aufgaben, wie die Erkennung von Sprache oder Objekten, zu erledigen (GOODFELLOW ET AL. 2016). Dies änderte sich durch

die Entwicklung von Expertensystemen hin zu datengetriebenen Modellen. Ein fundamentaler Unterschied besteht darin, dass diese Modelle aus Beobachtungen in Daten lernen, im Gegensatz zur Anwendung festgesetzter Formeln und Methoden (ROSCHER UND DREES, 2022). Das selbstständige Lernen aus vorhandenen Daten bedeutet jedoch keineswegs, dass das Modell nur rational entscheidet. Entscheidend sind hier die durch den Programmierenden vorgegeben Daten, wodurch die KI auch unmoralische Entscheidungen treffen kann. Die Intelligenz von diesen schwachen KI-Systemen bezieht sich ausschließlich auf die Vorgabe der großen Datenmengen mit zugehörigen Labels durch den Programmierenden. Das Modell lernt somit nachfolgend Muster und Zusammenhänge auf Basis der verwendeten Daten. Diese datengetriebenen Ansätze fallen in die Teilbereiche des Machine Learnings und des Deep Learnings.

2.2 Machine Learning

Beim Machine Learning (ML) als Teilbereich der künstlichen Intelligenz sollen aufgesetzte Systeme die Möglichkeit erhalten, selbstständig aus den Eingangsdaten zu lernen. Durch statistische Methoden verbessert das Modell stetig seine Genauigkeit und gelangt dadurch zu einem besseren Ergebnis. Die Wahl der Eingangsdaten bestimmt somit maßgeblich, was das Modell innerhalb des Lernprozesses, dem sog. Training, lernt. Mit dem Erlernten wird das Modell befähigt Zusammenhänge und unbekannte Muster innerhalb der Daten zu erkennen, die dem Menschen oftmals verborgen bleiben (LUTTKE, 2020). Durch die Anwendung auf unbekannte Testdaten kann die Vorhersagegenauigkeit des Modells ermittelt, weiterentwickelt und verbessert werden. Da es sich beim ML um einen datengetriebenen Prozess handelt, wird eine Vielzahl von Eingangsdaten benötigt, um eine gute Vorhersagegenauigkeit zu ermöglichen. Versteht man einen Datensatz als eine Ansammlung von Instanzen n (Zeilen des Datensatzes) mit einer Anzahl von d Attributen/Merkmalen (Spalten des Datensatzes), so trägt jedes weitere Attribut zur Erhöhung der Rechenzeit (n^d) bei. Weitere Attribute können als zusätzliche Dimensionen gesehen werden. Bei großen Datenmengen werden deshalb vor dem Training Dimensionsreduktionen durchgeführt, um dem Fluch der Dimensionalität zu entgehen (VERLEYSEN & FRANÇOIS 2005). Eine klassische Möglichkeit zur Dimensionsreduktion ist die Hauptkomponentenanalyse („principle component analysis“ (PCA)). Mittels PCA lässt sich eine Repräsentation des originalen Datensatzes mit niedriger Dimensionalität erstellen. Prinzipiell geht es um die Eliminierung stark korrelierender Attribute, deren Präsenz nicht

zur besseren Erklärung der Varianz der originalen Daten beiträgt. Zusätzlich werden nicht-lineare Beziehungen zwischen den Variablen eliminiert (GOODFELLOW, 2016). Als Ergebnis erhält man eine Repräsentation der Daten, die trotz niedriger Dimensionen den nahezu vollen Informationsgehalt der Originaldaten enthält. Weiterhin definiert die Struktur der Daten sowie das Lernziel den Lernprozess, der angewandt werden kann. Innerhalb des maschinellen Lernens unterscheidet man daher zwischen unüberwachtem, überwachten und verstärktem Lernen (ROSCHER UND DREES, 2022). Abbildung 1 gibt einen Überblick über die Teilbereiche des MLs. Beim überwachten Lernen stehen gelabelte Beispieldaten zur Verfügung, wodurch im Voraus schon feststeht, wie das Ergebnis und die damit verbundene Ausgabe aussehen sollten. Das Ziel ist durch die Trainingsdaten Strukturen und Regeln zu erkennen, die die Eingangsdaten mit der Ausgabe in Verbindung bringen (KIRSTE & SCHÜRHOLZ, 2019). Auf Basis dieser Erkenntnis soll das Modell befähigt werden Vorhersagen über unbekannte Daten zu machen. Klassische Verfahren des überwachten Lernens sind die Regression und die Klassifikation (vgl. Abb. 1). Bei der Regression geht es um die Ermittlung einer Funktion die bspw. einen linearen Zusammenhang zwischen einer unabhängigen Eingangsvariable und einer abhängigen Ausgangsvariable bestimmt. Als Beispiel könnte hier eine Stromverbrauchsprognose herangezogen werden. Als Ergebnis soll ein Modell erzeugt werden, dass auf Grundlage der Eingangsdaten den kontinuierlichen Output schätzen soll. Bei der Klassifikation hingegen werden die Eingangsdaten durch die Label definierten Klassen zugewiesen. Im Gegensatz zur Regression handelt es sich hier somit um eine kategorische Ausgabe, wo das Modell ungesehene Daten bekannten Klassen zuweisen soll (ROSCHER UND DREES, 2022). Prominente Klassifikationsaufgaben sind die Objekt- und Texterkennung (vgl. Abb. 1). Bei der Objekterkennung werden Objekte mit einer ähnlichen Semantik gleichen Klassen zugeordnet. Die gelabelten Trainingsdaten legen dabei fest, welche Objekte der Algorithmus erkennen kann. Die erkannten Objektinstanzen werden in der Regel mit sog. Bounding Boxen als Ergebnis hervorgehoben (ROSCHER UND DREES, 2022). Heutzutage werden solche Objekterkennungen eher mit Deep Learning Algorithmen durchgeführt,

da diese architekturbedingt bessere Ergebnisse liefern (Kapitel 2.3) (GOODFELLOW ET AL. 2016).

Im Gegensatz zum überwachten Lernen sind beim unüberwachten Lernen keine gelabel-

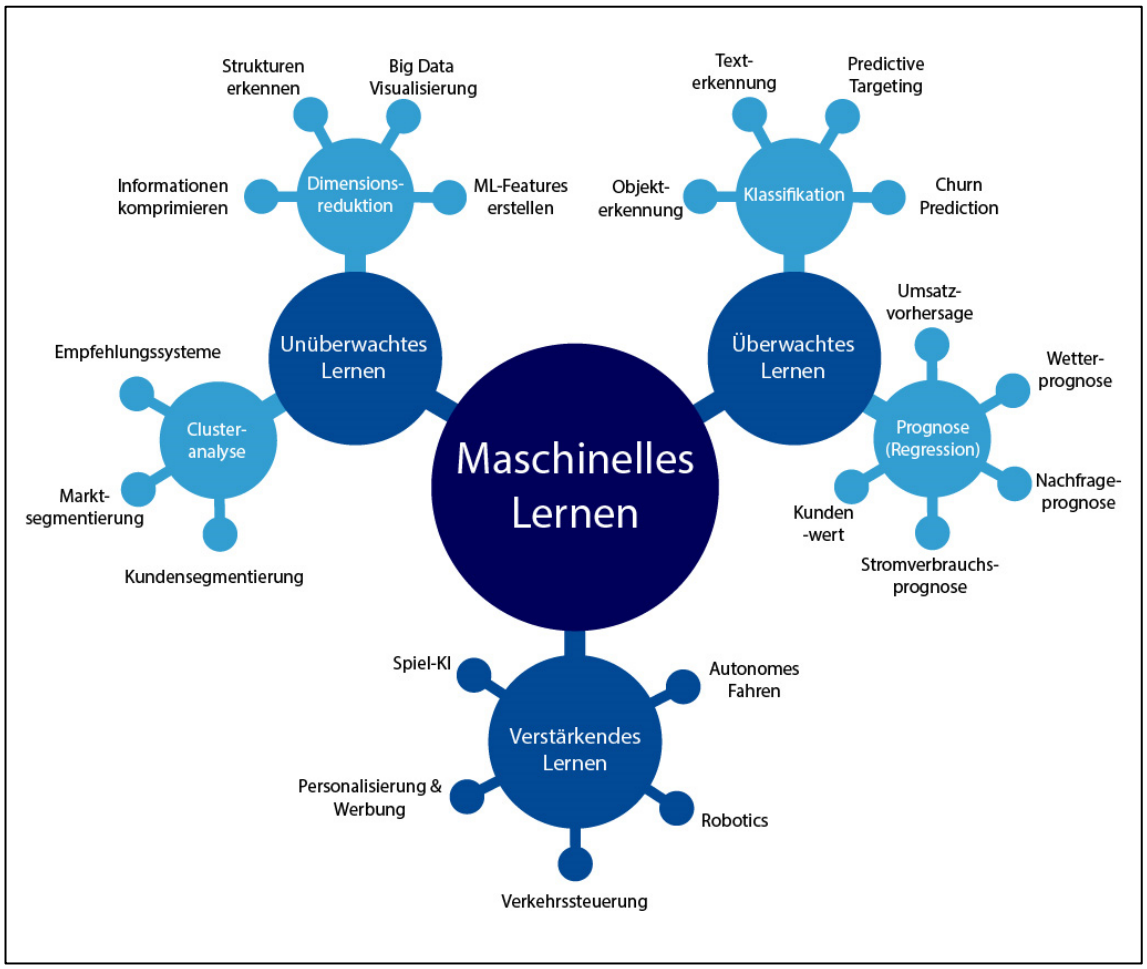


Abbildung 1: Teilbereiche des maschinellen Lernens mit beispielhaften Anwendungsfeldern für die einzelnen Teilbereiche (Quelle: LUTTKE, 2020)

ten Daten notwendig. Der Algorithmus bekommt somit nicht vom Programmierenden vorgegeben, nach welchen Strukturen und Inhalten er im Datensatz suchen soll. Viel eher geht es hierbei um das Erkennen von Inhalten, die dem Menschen nicht oder nur durch hohen Aufwand ersichtlich sind. Zur Verdeutlichung hilft die Vorgehensweise beim Clustering als Verfahren des unüberwachten Lernens. Beim Clustering ist das Ziel Ähnlichkeiten in den Daten zu finden, um diese nahezu homogenen Gruppen zuzuweisen (SEMMELMANN. 2021). Solche unüberwachten Lernverfahren begleiten uns im Alltag sehr häufig, gerade wenn es um Kunden- oder Marktsegmentierungen geht, beispielsweise bei zugeschnittener Werbung im Onlinehandel (KIRSTE & SCHÜRHOLZ, 2019).

Als letzter Teilbereich neben dem überwachten und unüberwachten Lernen bleibt das verstärkte Lernen. Anders als bei den ersten beiden Bereichen, wo dem Programm zu

Beginn ein Datensatz zum Lernen übergeben wird, ist dies hier nicht der Fall. Beim verstärkten Lernen soll das Programm durch Interaktion mit seiner Umgebung lernen. Es sammelt Erfahrungen, und wird für richtige Ergebnisse belohnt. Durch diese Belohnungen lernt es sich anzupassen und die eigene Belohnung zu maximieren. Ein bekanntes Beispiel aus dem letzten Jahrzehnt ist AlphaGo Zero. Hierbei handelt es sich um ein Programm der Firma „Google Deep Mind“, welches das Spiel GO ohne vorherige Kenntnisse erlernte. Das Ergebnis war so gut, dass es bereits nach drei Tagen in der Lage war die weltbesten menschlichen Spieler zu schlagen (SILVER ET AL. 2017, KIRSTE & SCHÜRHOLZ, 2019). Verstärktes Lernen ist besonders für den Bereich der Robotik sowie des autonomen Fahrens zukunftweisend, da hier das Verhalten zur Interaktion mit der eigenen Umwelt optimiert werden soll (vgl. Abb. 1).

Ein weiterer Aspekt, der bei den verschiedenen Teilbereichen des MLs nicht vernachlässigt werden darf, ist der Zeitpunkt des Lernens. In diesem Zusammenhang gibt es eine Unterscheidung zwischen Offline- und Online-Learning. Beim Offline Learning lernt das System zuerst aus den Daten und wendet seine gemachten Erfahrungen anschließend auf einen Anwendungsfall an. Beim Online-Learning werden hingegen die gemachten Erfahrungen angewandt und stetig das Verhalten angepasst, um dem Anwendungsszenario gerecht zu werden. Die Trennung zwischen Training und Anwendung erfolgt hier nicht.

Zusammengefasst geht es in allen genannten Teilbereichen um das Sammeln von Erfahrungen, um die zu lösende Aufgabe möglichst optimal zu bewältigen. Erfahrungen können aus bereitgestellten Trainingsdaten (unüberwachtes und überwachtes Lernen) oder durch Interaktion mit der Umgebung (Verstärktes Lernen) generiert werden. Wie MITCHELL bereits 1997 definierte: „A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, it its performance at task in T, as measured by P, improves with experience E“. Durch dieses selbstständige Lernen aus gesammelten Erfahrungen ist es möglich geworden Aufgaben zu bearbeiten, die sowohl feststehende Programme als auch der Mensch nur schwer bewältigen konnten (GOODFELLOW ET AL. 2016).

2.3 Deep Learning

Deep Learning (DL) ist die zukunftsweisende Technologie, die im Bereich der Erdbeobachtung wachsenden Einzug findet (CAMPS-VALLS ET AL. 2021). Als Teilbereich des MLs versucht man beim DL methodisch die Informationsverarbeitung des menschlichen Gehirns nachzubilden. Das menschliche Gehirn nutzt hierfür Neuronen, die über ihre Synapsen mit unzähligen weiteren Neuronen kommunizieren. Durch die starke Verschachtelung erhält ein einzelnes Neuron kontinuierlich Impulse anderer Neuronen und lädt sich auf, bis ein gewisser Schwellenwert erreicht ist. Im Anschluss gibt dieses Neuron einen Impuls über sein Axon und letztendlich der anschließenden Synapsen an weitere nachgeschaltete Neuronen ab (SCHÜRHOLZ & SPITZNER, 2019). Vergleicht man nun biologische neuronale Netze mit den künstlichen neuronalen Netzen (KNNs), die die Basis des DLs darstellen, so wird dieser komplexe Prozess eher abstrahiert. KNNs erhalten wie das menschliche Gehirn Informationen jeglicher Art als Eingangswerte. Diese werden verarbeitet und letztlich in Form eines Ausgangswertes interpretiert. Ein wesentlicher Aspekt liegt hier auf den Eigenschaften der Verschachtelungsstärke /Gewichtung und des Schwellenwerts, die beiden Netzen zugrunde liegen (SCHÜRHOLZ & SPITZNER, 2019; siehe Kap 2.3.1).

Obwohl man bereits seit 1952 einzelne Neuronen und deren Funktionsweise mittels dem Hodgkin-Huxley-Modell simulieren kann, gewinnt das DL erst seit ca. zehn Jahren an Bedeutung (KIRSTE & SCHÜRHOLZ, 2019). Dies wird vor allem durch die zunehmende Verfügbarkeit von Eingangsdaten und verbesserte Rechenleistung der Hardware bedingt (ZHANG ET AL. 2021). In den Anfängen des DLs wurden KNNs mit sogenannten Central Processing Units (CPUs) berechnet. CPUs sind zentrale, gemeinsame Recheneinheiten zur Durchführung komplexer Aufgaben wie bspw. aufeinander aufbauende Berechnungen (SCHÜRHOLZ & SPITZNER, 2019:36). Diese werden dort sequenziell abgearbeitet, was für viele Anwendungen äußerst effizient ist. Für das DL hingegen eignen sich diese Systeme weniger, da dort keine komplexen Berechnungen, sondern eine Vielzahl einfacher Matrixmultiplikationen durchgeführt werden müssen. Trotz der Einführung von Multithreading können diese Prozessoren bei parallelen Rechenoperationen nicht optimal ausgelastet werden. Aus diesem Grund werden seit einiger Zeit „Graphics Processing Units“ (GPUs) für solche Aufgaben verwendet. Im Verhältnis zur CPU wurden GPUs in der jüngsten Vergangenheit stark weiterentwickelt und in der Leistung gesteigert. GPUs ba-

sieren auf sogenannten Shader (kleine Rechenkerne für bestimmte Aufgaben im Grafikbereich). Früher hatten diese Shader festgesetzte Aufgaben und berechneten somit bspw. die Farbe oder Geometrie einzelner Bildpunkte. Mit der Einführung der Unified Shader-Architektur entfiel die vorherige Zuweisung bestimmter Aufgaben und es entstanden kleine Universalprozessoren (SCHÜRHOLZ & SPITZNER, 2019:39). Diese sind in der Lage, verschiedene Aufgaben je nach Bedarf zu erledigen. Dadurch wurde die GPU abseits der reinen Bildverarbeitung nutzbar und durch die effiziente Parallelisierung für das DL unabdingbar.

Die Bedeutung des Einsatzes von GPUs im DL bei steigender Verfügbarkeit von Daten wird in Tabelle 1 gezeigt. ZHANG ET AL. (2021) bedienen sich hier der Kenngröße „Floating Point Operations Per Second“ (FLOPS) zur Bewertung der Performance verschiedener Hardware beim Training von wachsenden Datensätzen. Während der Iris Datensatz in den 1970er mit einem Intel8080 CPU mit 100KF (100×10^3) trainiert wurde, erreichte man in den 2020er mit der NVIDIA DGX-2 (GPU) bereits einen PetaFLOP (1×10^{15}).

Tabelle 1: Bewertung der Performance mittels „Floating Point Operations Per Second (FLOPS) (modifiziert nach ZHANG ET AL. 2021).

Dekade	Datensatz	FLOPS
1970	100 (Iris)	100 KF (Intel 8080)
1980	1K (Häuserpreise in Boston)	1 MF (Intel 80186)
1990	10K (Optische Buchstabenerkennung)	10 MF (Intel 80486)
2000	10M (Homepages)	1 GF (Intel Core)
2010	10G (Werbung)	1 TF (Nvidia C2050)
2020	1T (Soziale Netzwerke)	1 PF (Nvidia DGX-2)

Tabelle 1 verdeutlicht die wachsende Verfügbarkeit von Daten und die notwendige Performancesteigerung zu Bewältigung dieser Datenmengen. Gerade auch durch den Einsatz der GPUs werden diese Berechnungen mit komplexeren Technologien möglich. Eine solche Technologie sind die „Convolutional Neural Networks (CNNs)“ (LECUN ET AL. 1999), die die Basis der Auswertungen innerhalb dieser Arbeit stellen.

2.3.1 Deep Feedforward Networks

Deep Learning Modelle werden in der Basis häufig als sogenannte „Deep Feedforward Networks (DFNs)“ oder auch „Multilayer Perceptrons (MLPs)“ bezeichnet. Strukturell wird zwischen verschiedenen Schichten bzw. Layern unterschieden. Neben den Input Layern, die die Eingangsdaten repräsentieren, gibt es sog. versteckte Layer (engl. Hidden layers) und den Output Layer. Der Output Layer stellt hierbei den Modelloutput dar, wohingegen in den versteckten Layern eine Kette von mathematischen Operationen durchgeführt wird. Ein Beispiel sind Filteroperationen auf Pixelebene, wie bei den CNNs (DERU & NDIAYE, 2019:69). CNNs sind eine spezielle Art der Feedforward Networks (vgl. Kap. 2.3.2). Von Feedforward spricht man in diesem Zusammenhang aufgrund des Informationsflusses durch das Netz. Vom Input Layer werden Eingangsdaten in den versteckten Layern (versteckt, da diese nicht ausgegeben werden) verrechnet und als Output klassifiziert. Der Modelloutput wird nicht in das Netz als Feedback zurückgegeben. Werden DFNs erweitert und solche Feedback Connections implementiert, spricht man von „Recurrent Neural Networks (RNNs)“. Die Anzahl der versteckten Schichten definiert die Tiefe der Netze. Ist mehr als eine versteckte Schicht vorhanden, spricht man von einem tiefen neuronalen Netz (DERU & NDIAYE, 2019:72).

DFNs sind leistungsstarke Modelle, die mittels der Anpassung der Gewichte w_n jedes Neurons versuchen vorgegebene Labels y der Trainingsdaten x_n zu reproduzieren. Während des Trainings innerhalb der DFNs wird nun die lineare Funktion $y = \Sigma w_n x_n + b$ gelernt, die Gewichte ständig angepasst und mit dem erwarteten Ergebnis verglichen. Dies erfolgt iterativ so lange, bis die Funktion die erwarteten Ausgaben optimalerweise bestimmt, bzw. so gut wie möglich annähern kann. Abb. 2 zeigt dieses Vorgehen schematisch für ein Neuron. Nach ROSCHER UND DREES (2022) kann deshalb „jedes Neuron nach dem Input Layer als Funktion angesehen werden, dessen Funktionswert vom Input, der Gewichtung des Inputs und dem Bias b abhängt“. Der Funktionswert kann als Aktivierung bezeichnet werden und wird durch Anwendung einer nichtlinearen Aktivierungsfunktion von einem linearen Ergebnis in ein nichtlineares Ergebnis geändert. Diese Anpassung ermöglicht dem Netz die Erkennung von Mustern, die sonst verborgen geblieben wären (NWANKPA ET AL. 2018, ROSCHER UND DREES 2022). Je nach Anwendungsfall wird zwischen verschiedenen Aktivierungsfunktionen unterschieden (näheres im weiteren Verlauf dieses Kapitels, im Abschnitt Aktivierungsfunktionen).

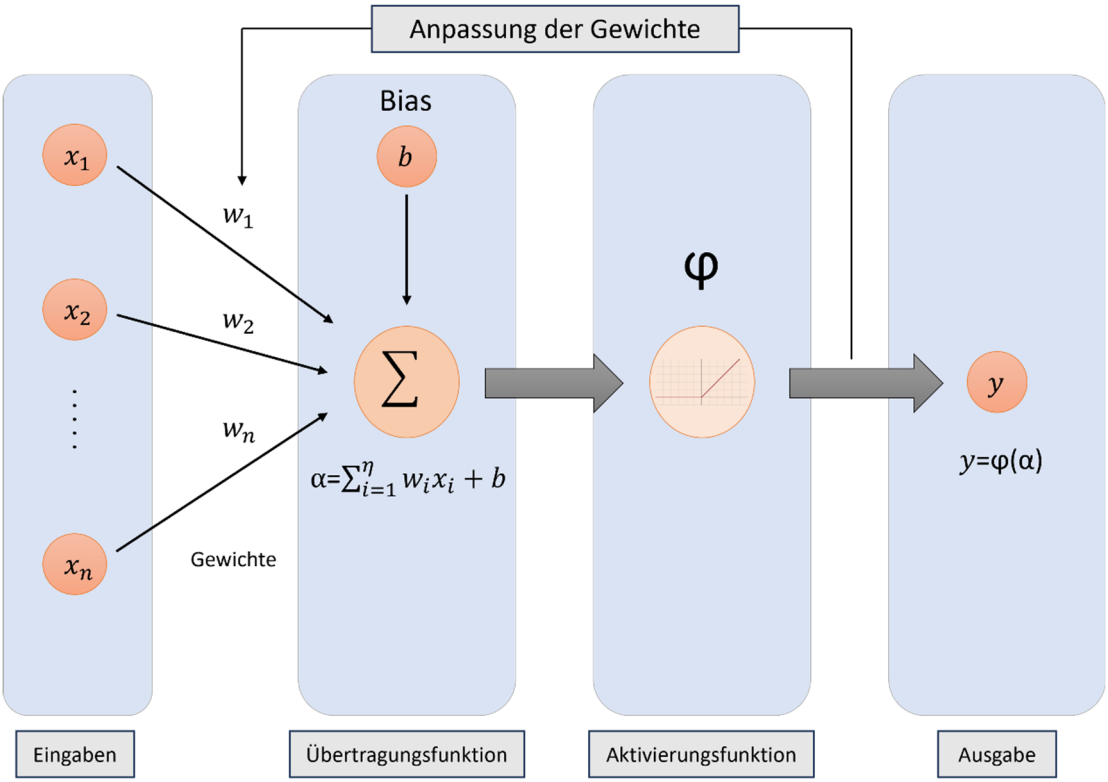


Abbildung 2: Schematische Darstellung der Gewichtsanpassung eines Neurons innerhalb des Trainingsprozesses eines neuronalen Netzes (modifiziert nach DERU & NDIAYE, 2019:71)

Generalisierung

Die Generalisierung ist eine fundamentale Herausforderung bei der Arbeit mit DL-Ansätzen. Ziel ist es, mit dem trainierten DL-Modell, hohe Treffergenauigkeiten auf ungesehenen Daten zu erzielen. Während des Trainings wird mittels Gradientenabstiegsverfahren (vgl. Abschnitt Gradientenabstieg) versucht, die Verlustfunktion (vgl. Abschnitt Verlustfunktion) zu minimieren. Dies kann allerdings dazu führen, dass sich das Modell zu stark an die Trainingsdaten anpasst, und diese auswendig lernt. Hierbei spricht man von Overfitting. Das Modell kann die Varianz der Trainingsdaten bis ins Detail abbilden, erkennt aber nicht die grundlegenden Muster, um auf ungesehenen Daten hohe Treffergenauigkeiten zu erzielen. Würden man dieses Modell auf Testdaten anwenden, würde man mit hoher Wahrscheinlichkeit wesentlich höheren Testverlust oder auch Generalisierungsfehler wahrnehmen. Um dies zu vermeiden, gibt es verschiedene Ansätze, die als Regularisierungen bezeichnet werden (GOODFELLOW ET AL. 2016:116F). Regularisierungen zielen auf die Verminderung der Modellkomplexität ab. Durch Verfahren wie L1 – und L2- Regularisierung nehmen die Gewichtsparameter geringere Werte an, wodurch

der Einfluss einzelner Neuronen aus den versteckten Schichten reduziert wird. Dies erfolgt über eine Modifikation der Verlustfunktion durch Hinzufügen eines Regularisierungsterms α (L1 oder L2) (GOODFELLOW ET AL. 2016:226FF). Für die Wahl von α gilt es die Balance zwischen Genauigkeit und Komplexität des Modells zu beachten. Wird zu stark regularisiert, ist zwar die Modellkomplexität und die Gefahr des Overfittings verringert, jedoch ist das Modell gegebenenfalls nicht in der Lage zu generalisieren. Der Lernprozess aus den Trainingsdaten ist nicht ausreichend, um genaue Vorhersagen mit hoher Genauigkeit zu treffen. Ist α hingegen zu gering, bleibt die komplexe Modellstruktur erhalten und es werden zu viele Details aus den Trainingsdaten gelernt. Die Schlussfolgerung sind niedriger Trainingsverlust und hoher Testverlust. Nach GOODFELLOW ET AL. (2016:109) sind performante Modelle in der Lage, sowohl den Trainingsverlust als auch die Differenz zwischen Trainings- und Testverlust klein zu halten. Erreicht man sowohl für das Training als auch für den Test ähnliche Verluste, generalisiert das Modell adäquat. Sollten beide Genauigkeiten allerdings sehr gering sein, kann Underfitting ein Problem sein. Hier ist die Komplexität des Modells zu gering, um die Daten zu beschreiben, und den Trainingsverlust auf ein globales Minimum (vgl. Abschnitt Gradientenabstieg) zu senken. Dieses Phänomen tritt beispielsweise bei einer zu geringen Epochenanzahl oder zu wenig Eingangsdaten während des Trainingsprozesses auf. Hier können Verfahren der „Data Augmentation“ hilfreich sein, bei denen künstlich Abbilder der Eingangsdaten generiert werden. Diese sind je nach Anwendung beispielsweise rotiert oder horizontal und vertikal verschoben (DERU & NDIAYE, 2019:458).

Neben der L1- und L2- Regularisierung gibt es noch den Dropout um die Generalisierung des Modells zu gewährleisten. Hierbei werden während des Trainings einzelne Neuronen durch Angabe eines Wahrscheinlichkeitswertes P entfernt. Dadurch ist die Modellstruktur weniger komplex und die Möglichkeit des Overfittings wird reduziert (DERU & NDIAYE, 2019:459).

Aktivierungsfunktion

Bei der Anpassung der Gewichte spielen Aktivierungsfunktionen eine entscheidende Rolle. Wie in Abb. 2 bereits dargestellt, werden in jedem Neuron der versteckten Schichten lineare Transformationen durchgeführt und abschließend evaluiert, inwieweit der vorhergesagte Wert mit dem erwarteten Wert übereinstimmt. Aktivierungsfunktionen tragen nun dazu bei, dass komplexere nicht-lineare Strukturen und Muster in dem Datensatz

erkannt werden können. Würde ein neuronales Netz keinerlei Aktivierungsfunktionen erhalten, hätte man lediglich eine Art lineares Regressionsmodell, welches nicht in der Lage wäre, komplexe Muster zu erkennen (NWANKPA ET AL. 2018). Im Laufe der Zeit haben sich in Bezug auf KNNs einige Aktivierungsfunktionen etabliert, die für verschiedene Anwendungen geeignet sind. Eine Auswahl wird in diesem Kapitel gezeigt:

Sigmoid-Funktion

Die Sigmoid Funktion ist eine der meistgenutzten Aktivierungsfunktionen. Sie besitzt graphisch eine markante S-Form mit einer Wertverteilung von 0 bis 1 und wird wie folgt definiert (SHARMA AND ATHAIYA, 2020):

$$f(x) = \frac{1}{1 + e^{-x}} \tag{1}$$

Die Sigmoid Funktion wird häufig in flachen Netzwerken zur Lösung binärer Klassifikationen oder Regressionen verwendet, da sie sehr verständlich ist. Die großen Nachteile von der Sigmoid Funktion sind langsame Konvergenz, Gradientensättigung und ein Ergebnis, dass nicht symmetrisch um null zentriert ist. Betrachtet man den Graphen in Abb. 3, nimmt man den sehr flachen Kurvenverlauf der Sigmoid Funktion in der Nähe von 0 und 1 wahr. Hier ist die Ableitung für sehr große und sehr kleine Eingangswerte nahe null, was negative Auswirkungen auf den Gradienten der Verlustfunktion sowie die Aktualisierung der Gewichte hat. Dieses Problem wird „Vanishing Gradient Problem“ genannt (NWANKPA ET AL. 2018). Die fehlende Zentrierung um null hat Einfluss auf das Training des Netzes. Sie führt dazu, dass alle Gewichte in einem sigmoiden Neuron entweder erhöht oder verringert werden können, aber keine Unterscheidung für einzelne Gewichte innerhalb des Neurons gemacht werden kann (SHARMA AND ATHAIYA, 2020).

Tangens-Hyperbolicus-Funktion

Im Gegensatz zur Sigmoid-Funktion ist die Tangens-Hyperbolicus-Funktion (tanh) um null zentriert, was in einer besseren Anpassung der Gewichte resultiert. Jedoch kann die tanh-Funktion ebenfalls nicht das „Vanishing Gradient Problem“ lösen, hier im Wertebereich von -1 und 1 (vgl. Abb. 3). Eine Eigenschaft von tanh ist die Produktion sog. toter Neuronen. Sind die Eingangsdaten vom Wert null, ist der Gradient eins. Dies hat zur Folge, dass die Gewichte nicht angepasst und kein Training erfolgen kann. Mathematisch ist die tanh-Funktion wie folgt definiert:

$$f(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \tag{2}$$

Als Aktivierungsfunktion wird tanh eher in RNNs zur Erkennung von Sprache verwandt (NWANKPA ET AL. 2018, DAUPHIN ET AL. 2017).

Softmax Funktion

Die Softmax-Funktion wird gerade für die Ausgangsneuronen bei mehrklassigen Klassifizierungsaufgaben häufig verwendet. Mittels Softmax wird eine Wahrscheinlichkeitsverteilung über alle sich gegenseitig ausschließenden Klassen dargestellt. Hierbei werden den Werten der Ausgabeneuronen Wahrscheinlichkeiten im Intervall [0,1] zugewiesen (SHARMA AND ATHAIYA, 2020). Diese werden für jede Klasse ausgegeben, wobei die zugehörige Klasse die höchste Wahrscheinlichkeit aufweist. Aus diesem Grund ergeben die aufsummierten Wahrscheinlichkeiten den Wert 1. Die beschriebene Abhängigkeit wird in Formel 3 aufgezeigt:

$$f(x_i) = \frac{e^{x_i}}{\sum_j e^{x_j}} \tag{3}$$

Aus diesem Grund eignet sich Softmax bei multivariaten Klassifikationen besser als Sigmoid, die primär bei binären Klassifikationen verwendet wird (NWANKPA ET AL. 2018).

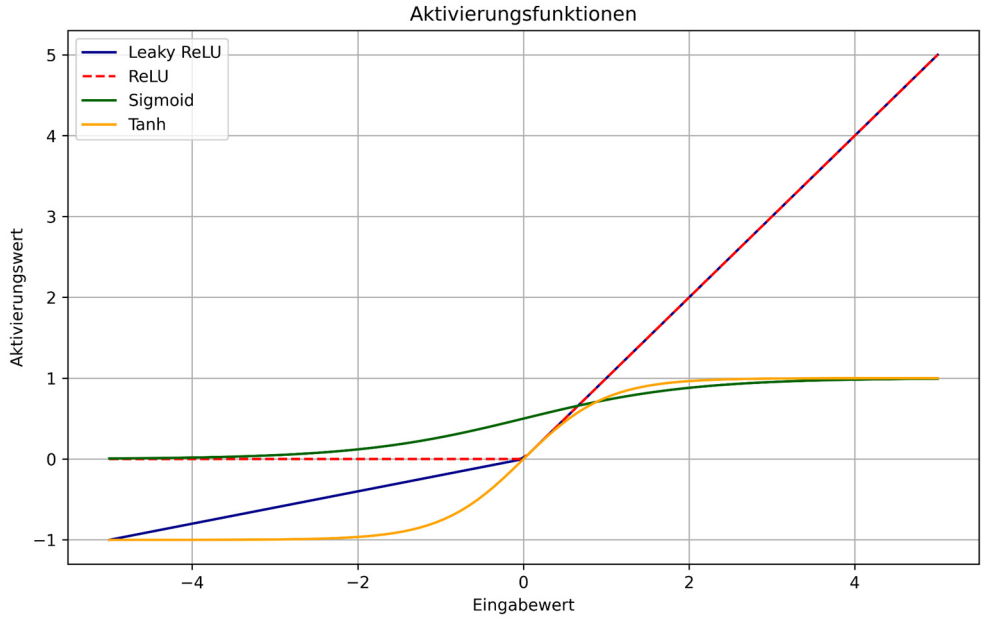


Abbildung 3: Graphische Darstellung gängiger Aktivierungsfunktionen. Dargestellt sind ReLU, Leaky ReLU, Sigmoid und Tanh (Eigene Darstellung).

Rectified-Linear-Unit Funktion (ReLU)

Seit der Einführung im Jahr 2010 durch NAIR UND HINTEN ist ReLU eine beliebte Aktivierungsfunktion beim Training von KNNs (NAIR UND HINTEN, 2010). ReLU zeichnet sich durch bessere Trainingsperformance im Gegensatz zu anderen Aktivierungsfunktionen aus, speziell in Bezug auf die Konvergenz zum globalen Minimum der Verlustfunktion (vgl. Abschnitt Gradientenabstieg). Die Formel 4 zeigt die einfache Implementierung (vgl. Abb. 3):

$$R(x) = max(0, x) = \begin{cases} x \text{ für } x \geq 0 \\ 0 \text{ für } x < 0 \end{cases} \quad (4)$$

Bei ReLU kommt es durch Schwellenwertoperation, wo alle Eingangswerte unter 0 gleich 0 gesetzt werden, unter Umständen zu sehr fragilen bis toten Neuronen. Führt die Anpassung der Gewichte zu negativen Gewichten, so sind die Ausgangswerte dieser Neuronen bei ReLU immer 0. So sind Gradienten, die durch diese ReLU-Neuronen fließen gleich null. Sie sterben und werden während des Trainings nicht mehr aktiviert (NWANKPA ET AL. 2018, SHARMA AND ATHAIYA, 2020).

Leaky ReLU

Bei Leaky ReLU handelt es sich um eine erweiterte Version von ReLU. Durch die Einführung eines α - Wertes werden tote Neuronen und damit verbundene Nullwerte bei den Gradienten verhindert (vgl. Abb. 3). Aus diesem Grund werden die Gewichte während des gesamten Trainings geupdatet. Im Grunde wird die horizontale Nulllinie bei ReLU durch Einführung des α-Werts zu einer linearen Linie (vgl. Abb. 3). Mathematisch wird Leaky ReLU wie folgt beschrieben:

$$R(x) = \begin{cases} x & \text{für } x \geq 0 \\ \alpha x & \text{für } x < 0 \end{cases} \quad (5)$$

Verlustfunktion

Die Wahl der Verlustfunktion ist für jede Netzarchitektur unabdinglich, da Sie Auskunft gibt, wie gut das Model die gestellte Aufgabe löst. Je nach Anwendungsfall wird zwischen verschiedenen Verlustfunktionen (engl. „loss function“) unterschieden. Bei Klassifizierungsaufgaben geht es um die Interpretation von Wahrscheinlichkeitsverteilungen. Eine Möglichkeit den Fehler zwischen vorhergesagter Wahrscheinlichkeit und der eigentlichen Klasse zu evaluieren, liefert die Cross-Entropy Verlustfunktion (siehe Formel 6):

$$l(\hat{y}_i, y_i) = -\sum_{i=0}^{N} \hat{y}_i \times log(y_i) \quad (6)$$

$\hat{y}_i$ stellt hier den Labelvektor dar, der die Zuordnung der richtigen Klasse durch den Wert 1 enthält. Für alle falschen Klassen enthält $\hat{y}_i$ eine 0. Neben dem Labelvektor gibt es noch den Vorhersagevektor y_i. Bei dem Cross Entropy Loss werden nun die Produkte des Labelvektors und des logarithmierten Vorhersagevektors aufsummiert und negiert. Die Negierung dient einer positiven Ausgabe der Verlustfunktion (GORDON-RODRIGUEZ ET AL. 2020).

Der Cross Entropy Verlust eignet sich zwar sehr gut für Klassifizierungsprobleme, allerdings ist die Anwendung darüber hinaus eingeschränkt. Wie eingangs erwähnt, gibt es eine Menge unterschiedlicher Verlustfunktionen, die sich für verschiedene Anwendungsfälle nutzen lassen. In diesem Abschnitt soll noch auf die Intersection over Union (IoU) als bekanntes Beispiel im Kontext der semantischen Segmentierung und auf den mittleren quadratischen Fehler („Mean Squared Error“ (MSE)) eingegangen werden. Der MSE wird häufig für Regressionsprobleme verwendet.

Bei Intersection over Union (IoU) handelt es sich um eine sehr bekannte Metrik zur Bewertung der Modelperformance. Gerade in Segmentierungen und Objektdetektionen wird diese häufig verwendet. Bei IoU wird die Überlappung von Vorhersage (P) und Originalbild / Originalklasse (T) bestimmt. Dabei wird die Schnittmenge von T und P durch deren vereinigte Menge geteilt (vgl. Formel 7).

$$IoU = \frac{|T \cap P|}{|T \cup P|} \quad (7)$$

$$IoU' = \frac{|T\ x\ P|}{|T + P - (T\ x\ P)|} \quad (8)$$

$$l_{IoU} = 1 - IoU' \quad (9)$$

Eine wichtige Eigenschaft von Verlustfunktionen ist deren Differenzierbarkeit. Formel 8 zeigt eine approximierte Version des IoU, die dieses Kriterium erfüllt (VAN BEERS ET AL. 2019). Diese muss anschließend noch von 1 subtrahiert werden (Formel 9). Die Subtraktion ist notwendig, um die Minimierung der Verlustfunktion als wichtigen Schritt der Backpropagation (siehe Abschnitt Training eines KNNs) zu gewährleisten. Umso besser die Vorhersage P mit dem Original T übereinstimmt, umso niedriger ist l_{IoU}.

Entgegen der Cross Entropy Funktion und IoU mit deren Interpretation von Klassenzugehörigkeiten geht es bei dem MSE und seiner Verwendung bei Regressionen um die Vorhersage kontinuierlicher Einzelwerte. Letztendlich werden quadrierte Differenzen zwischen Vorhersage (y) und den wahren Werten (ŷ) über alle Instanzen aufsummiert und durch die Anzahl der Instanzen (N) geteilt (SAMMUT UND WEBB, 2011:652):

$$l(\hat{y}_i, y_i) = \frac{1}{N}\sum_{i=0}^{N}(y_i - \hat{y}_i)^2 \tag{10}$$

Training eines künstlichen neuronalen Netzes

Das Training eines KNN besteht im Wesentlichen aus den folgenden drei Schritten (ROSCHER UND DREES, 2022):

- Vorwärtspropagation: Die Eingangsdaten werden in das Netz eingespeist, Aktivierungen innerhalb der versteckten Schichten und als Modelloutput final berechnet und ausgegeben.
- Berechnung der Verlustfunktion: Der Unterschied zwischen Vorhersage und der Zielvariable wird bestimmt. Dieser soll minimal sein.
- Backpropagation: Die Verlustfunktion wird nun durch ein iteratives Verfahren zur Anpassung der Modellgewichte durch das Gradientenabstiegsverfahren minimiert.

Aufgrund der direkten Abhängigkeit des Modelloutputs von der Anpassung der Modellgewichte sind deren Veränderungen maßgeblich für die Modellgüte. Ein sehr bekanntes Verfahren ist die Backpropagation mit Gradientenabstiegsverfahren (DERU & NDIAYE, 2019:77). Die Bestimmung einer neuen Gewichtung „ergibt sich aus der Differenz des alten Gewichts und der partiellen Ableitung des berechneten Fehlers E in Bezug auf die Gewichtung w_i“ (DERU & NDIAYE, 2019:77). Modifiziert wird dies durch die Lernrate η, die angibt, wie schnell das Modell Anpassungen vornehmen soll (siehe Formel 11). Weiterhin wird in unserem Fall mit sog. Minibatches M gerechnet, wo mittels Batchsize definiert wird, wie viele Trainingsdaten dem Netz auf einmal präsentiert werden sollen. Die Anpassung der Gewichte erfolgt hier am Ende einer Epoche.

$$w_{i_{new}} = w_{i_{old}} - \eta\, x\, \frac{1}{M}\sum_{k=1}^{M}\frac{\partial E}{\partial w_i} \tag{11}$$

Die Prozessierung in Batches trägt zur Robustheit des Trainings bei. Batchsize und Lernrate sind somit essenzielle Hyperparameter, die die Geschwindigkeit des Lernprozesses

stark beeinflussen. Solche Anpassungen werden nun rückkoppelnd von der letzten Schicht zur ersten Schicht durch das Netz durchgegeben und die Gewichtung angepasst (Backpropagation).

Wie in Formel 11 dargestellt, ist die Verlustfunktion E entscheidend bei der Backpropagation. Ziel ist es, diese so stark wie möglich zu minimieren und ein sog. globales Minimum zu finden. Im Deep Learning werden hierfür sogenannte Gradientenabstiegsverfahren bzw. die gradientenbasierte Optimierung verwendet.

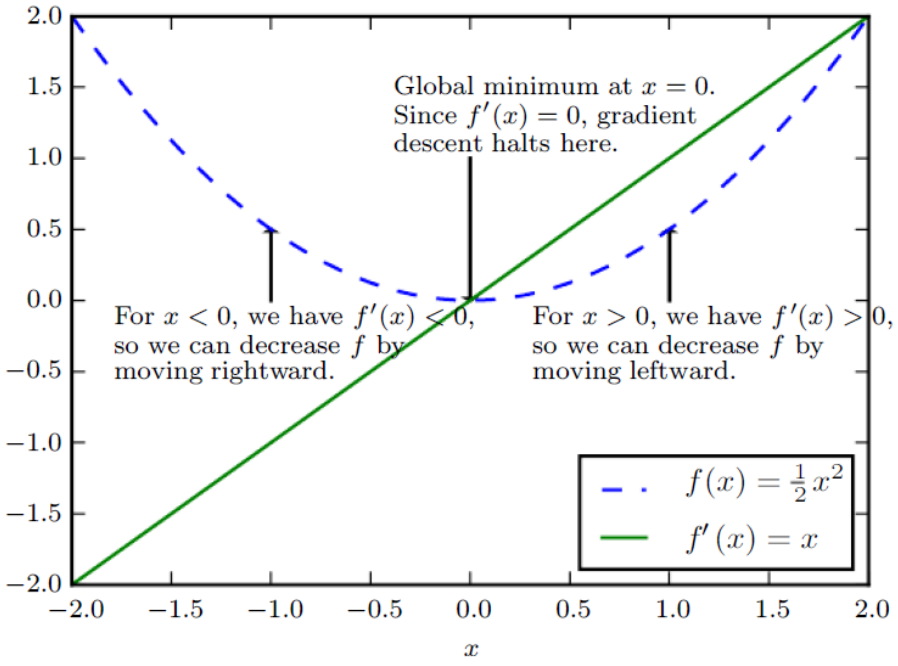


Abbildung 4: Die Rolle der Ableitung einer Funktion bei Gradientenabstieg. Illustriert durch die Funktion f(x) = 1/2 x² und deren Ableitung f‘(x) = x mit globalem Minimum bei x = 0 (GOODFELLOW ET AL. 2016:81).

Abbildung 4 illustriert das Konzept des Gradientenabstiegs (engl. Gradient descent (GD)). Zu sehen sind die zwei Funktionen $f(x) = 1/2x^2$ und deren Ableitung f‘(x) = x. Bei GD bedient man sich nun der Ableitung einer gegebenen Funktion f(x). Da die Ableitung f‘(x) eine Auskunft über die Steigung von f(x) in Punkt x gibt, lässt sich herleiten in welcher Richtung von f(x) ein sog. Minimum der Funktion zu finden ist. Schiebt man nun den Parameter der Verlustfunktion ein Stück in die entgegengesetzte Richtung der Steigung, nähert man sich dem Minimum an. Wie groß die Schritte der Anpassung sind, wird mittels Lernrate festgelegt. Der iterative Prozess wird fortgesetzt, bis die Steigung theoretisch den Wert null annimmt und optimalerweise ein globales Minimum erreicht wurde (in unserem Beispiel bei f(x) = 0). Neben dem globalen Minimum gibt es noch lokale Minima und Sattelpunkte, an denen die Steigung ebenfalls null ist. Um der alleinigen

Anpassung auf Basis des Gradienten und damit einhergehenden Problemen entgegenzuwirken, haben sich effektivere Optimierungsalgorithmen entwickelt. Bekannte Vertreter sind hier der Stochastische Gradientenabstieg (engl. Stochastic Gradient Descent (SGD)) mit Momentum und Adam (Adaptive Moment Estimation) (DERU & NDIAYE, 2019:80). Bei der Berechnung der Verlustfunktion beim GD fließen alle Trainingsdaten in die Berechnung mit ein, wodurch sich eine Rechenzeit bei einem n großen Datensatz von O(n) ergibt (Formel 12). Da hier die Gradienten des gesamten Datensatzes für jedes Update berechnet werden, ist mit langen Rechenzeiten zu rechnen (RUDER, 2017). Zur Optimierung der Rechenzeit, wird beim SGD für jede Iteration eine zufällige Instanz (Formel 13) oder ein bereits angesprochener Minibatch (Formel 14) gewählt. Dies führt zur Performancesteigerung aber gleichzeitig auch zu mehr Iterationen, da das Minimum nicht auf direktem Weg gefunden wird. Aufgrund von Parameteranpassung nach jeder genutzten Instanz fluktuiert die Verlustfunktion aufgrund vorhandener Varianz in den Daten sehr. Dies kann zwar zur Entdeckung neuer lokaler Minima führen, aber erschwert die Konvergenz zum globalen Minimum enorm (RUDER, 2017). Die Reduktion der Lernrate η sowie die Verwendung eines Minibatches führen zur stabileren Konvergenz. Mathematisch sind GD (Formel 12), SGD (Formel 13) und Minibatch SGD (Formel 14) für die Parameteraktualisierung wie folgt umgesetzt (hierbei sei η die Lernrate, θ die Parameter und L die Verlustfunktion):

$$\theta = \theta - \eta \times \nabla_\theta \mathrm{L}(\theta) \tag{12}$$

$$\theta = \theta - \eta \times \nabla_\theta \mathrm{L}(\theta; \mathrm{x}^{(\mathrm{t})}; \mathrm{y}^{(\mathrm{t})}) \tag{13}$$

$$\theta = \theta - \eta \times \nabla_\theta \mathrm{L}(\theta; \mathrm{x}^{(\mathrm{i:i+n})}; \mathrm{y}^{(\mathrm{i:i+n})}) \tag{14}$$

Um die Anzahl der Iteration beim SGD weiter zu verringern, wurde dieser um das Momentum erweitert. Beim SGD mit Momentum wird durch Hinzunahme von γ die Richtung zum Minimum stärker verfolgt. Das Momentum γ setzt sich anteilig aus dem letzten und dem aktuellen Updatevektor v_t (Formel 15) zusammen. Zeigen die Gradienten der Vektoren in die gleiche Richtung lädt sich das Momentum auf oder schwächt ab, bei entgegengesetzten Richtungen. Dieses „Aufladen“ führt zur geringeren Anfälligkeit für

spontane Richtungsänderungen, schnellerer Konvergenz und einer geringeren Anzahl an Iterationen (RUDER, 2017).

$$v_t = \gamma v_{t-1} - \eta \nabla_\theta \mathrm{L}(\theta; \mathrm{x}^{(t)}; \mathrm{y}^{(t)}) \tag{15}$$

$$\theta = \theta - v_t \tag{16}$$

Für weitere Verbesserungen im Training wurden adaptive Algorithmen entwickelt. Diese unterschieden sich im Umgang der Parameter, speziell wann und in welcher Form die Anpassung von Lernrate und Gewichten im Trainingsprozess erfolgt (DERU & NDIAYE, 2019:80). Bekannte Verfahren sind Adagrad, AdaDelta, RMSProp und Adam. Mit Adagrad als erstes Verfahren kommt es zu kontinuierlichen Anpassungen der Lernrate, wobei steile Gradienten größere Anpassungen als flache Gradienten erhalten. Problematisch bei Adagrad sind stark fallende Lernraten, die den Trainingsprozess hindern bis lahmlegen können. Um sich diesem Problem anzunehmen, wurden AdaDelta und RMSProp entwickelt. Wichtige Veränderungen waren die Einführung von quadrierten Gradienten und ein exponentieller Zerfall des Momentvektors. Im Gegensatz zu Adagrad haben nun nur die letzten Gradienten einen Einfluss auf den Momentvektor und nicht alle vorherigen (RUDER, 2017).

Adam („Adaptive Moment Estimation“) kombiniert nun RMSProp und das Momentum (KINGMA UND BA. 2014). Dies führt zur Verwendung des Durchschnitts der vorherigen (Momentum) v_t und der quadrierten (RMSProp) Gradienten m_t mit exponentiellem Zerfall (Formel 19).

$$\widehat{m}_t = \frac{ß_1 m_{t-1} + (1 - ß_1) g_t}{1 - ß_1^t} \tag{17}$$

$$\hat{v}_t = \frac{ß_2 v_{t-1} + (1 - ß_2) g_t^2}{1 - ß_2^t} \tag{18}$$

$$\theta_{t+1} = \theta - \frac{\eta}{\sqrt{\hat{v}_t} + \varepsilon} \widehat{m}_t \tag{19}$$

Für beide Momentvektoren $\widehat{m}_t$ und $\hat{v}_t$ gibt es vorher festgesetzte Zerfallsraten ($ß_1$ (0,9) und $ß_2$ (0,999)) mit welchen $\widehat{m}_t$ und $\hat{v}_t$ während des Trainings bei jeder Iteration neu bestimmt werden. Anschließend kommt es zum Update der Parameter θ unter Hinzunahme der Lernrate η und dem Parameter ε, welcher der Division durch 0 entgegenwirken

soll (Formel 19). Für die Konstante ε wird typischerweise 10^{-8} angenommen (RUDER, 2017; GOODFELLOW ET AL. 2016:306). Adam ist ein häufig verwendeter Optimierer im Deep Learning Kontext. Die Praxis zeigt, dass ohne große Anpassungen der Parameter gute Trainingsergebnisse erzielt werden können.

2.3.2 Convolutional Neural Networks

Der Unterschied zwischen klassischen KNNs und „Convolutional Neural Networks (CNNs)“ ist deren Netzarchitektur sowie deren Anwendungsfeld. Während wir bei KNNs über vollständig verknüpfte Schichten mittels Matrixoperationen zum Ergebnis gelangen, besitzen CNNs eine Operation namens Faltung (engl. Convolution). Bisher waren alle Neuronen einer Schicht mit allen Neuronen der vorherigen Schicht verbunden. Die Eingangswerte konnten so von der ersten bis zur letzten Schicht durchgegeben und durch iterative Anpassungen der Gewichte, eine Funktion zur Lösung der Aufgabe angenähert werden (Kapitel 2.3.1). Hierfür wurden dem Netz beim überwachten Lernen gelabelte Daten präsentiert, wovon das Netz Informationen lernen sollte, um auf ungesehene Daten angewendet zu werden. Hierbei war die Ordnung der Daten nicht entscheidend (HOESER UND KUENZER, 2020). Dies ist bei CNNs anders. CNNs sind vorwärtsgerichtet und durch ihre Architektur besonders für Klassifikationsaufgaben im Bereich der Bilderkennung geeignet (DERU & NDIAYE, 2019:91). Nach HOESER UND KUENZER (2020) sind Bilder Aufnahmen natürlicher Signale, die Informationen als Pixelwerte und deren Verbindung liefern. Klassische KNNs wären nicht in der Lage diese lokalen Pixelarrangements zu extrahieren. Dies passiert bei CNNs in den Faltungsschichten mittels Filteroperationen (DERU & NDIAYE, 2019:92).

Generell kann die Architektur eines klassischen CNNs in drei Teile eingeteilt werden: Die Eingangsschicht, den „convolutional backbone“ und den „classifiers head“. In der Eingangsschicht fließen die Eingangsdaten als 2D Array in das Netz. Anschließend werden im Backbone durch eine Reihe von Faltungs-, Aktivierungs- und Max Poolinglayer Bildeigenschaften extrahiert und im Classifier den Ausgangsklassen mit einer bestimmten Wahrscheinlichkeit zugewiesen (HOESER UND KUENZER, 2020).

Die Basis der CNNs bildet die sog. Faltungsschicht (engl. Convolutional Layer). Während der Faltung iterieren verschiedene Filter (engl. Kernel) über das Eingabebild, um Merkmale aus diesem zu extrahieren. Für ein zweidimensionales Grauwertbild lässt sich die Faltung mathematisch wie folgt ausdrücken (GOODFELLOW ET AL. 2016:329):

$$S(i,j) = (I\ x\ K)(i,j) = \sum_m \sum_n I(m,n)K(i-m,j-n) \tag{20}$$

Die Pixelwerte des Bildes I werden hier iterativ mit den Filterwerten K multipliziert und als Feature Maps S ausgegeben. Hierbei bestimmen Filtergröße (engl. Kernel Size) und Schrittweite (engl. Stride) die ausgegebene Feature Map. Um die abstrakte Formel 20 und das Konzept besser zu verstehen, zeigt Abb. 5 die praktische Anwendung der Faltung. Zu sehen ist ein 6x6 Eingabebild, worüber ein 3x3 Kernel iteriert, um eine 4x4 Feature Map zu erstellen. Beispielhaft stellen die grauen Pixel eine solche Filteroperation. Elementweise werden die einzelnen Pixel wie folgt multipliziert:

$$5 \text{ x } (-1) + 5 \text{ x } (-2) + 5 \text{ x } (-1) + 15 \text{ x } 0 + 15 \text{ x } 0 + 15 \text{ x } 0 + 7 \text{ x } 1 + 6 \text{ x } 2 + 7 \text{ x } 1 = 6$$

Anschließend springt der Filter um die angegebene Schrittweite (hier eins) nach rechts weiter und es werden die nächsten Pixel verrechnet. Iterativ werden so die Pixelwerte der Ausgabe berechnet. Sollte ein Reduzierung der Bildgröße von Eingabe zu Ausgabe nicht erwünscht sein, empfiehlt sich ein Padding. Hier werden dem Eingabebild Pixel (i.d.R. vom Wert null) hinzugefügt, um zu verhindern, dass die Bildgröße der Ausgabe schrumpft.

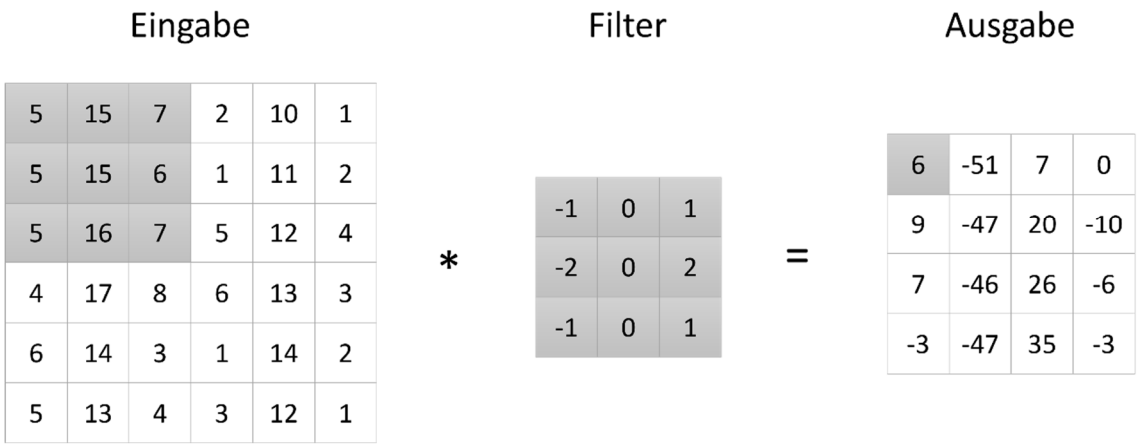


Abbildung 5: Beispiel einer Faltungsoperation. Das Eingaberaster (6x6) wird mit einem 3x3 Filter gefaltet und resultiert in einer 4x4 Ausgabe. Hierbei wird mit einer Schrittweite von 1 über die Eingabe iteriert (Eigene Darstellung).

Innerhalb eines CNNs werden nun sehr viele verschiedene Filter eingesetzt. So gibt es bspw. Weichzeichner oder Schärfungsfilter, Kantenfilter (wie Sobel in Abb. 5), Mittelwertfilter et cetera. Das Netz entscheidet hier nach Anpassung der Gewichte selbst, welche Feature Maps stärker und schwächer einfließen, je nachdem welche Merkmale es aus den Daten liest (DERU & NDIAYE, 2019:93).

Die Faltungsoperation weist weiterhin entscheidende Vorteile in Bezug auf Rechenzeit und notwendigen Speicher auf. Durch „Sparse Connectivity“ und Parameter sharing werden sowohl Rechenzeit als auch der notwendige Speicher reduziert (GOODFELLOW ET AL. 2016:329FF). Durch die Wahl eines kleineren Filters ist man in der Lage, selbst bei großen Eingabebildern, feine Strukturen im Bild zu erkennen und benötigt je nach Filtergröße weniger Gewichte. Das Eingabebild und das Ausgabebild besitzen hier eine „Sparse Connectivity“, da der Filter hier die Vernetzung angibt, anders als beim klassischen KNN (vgl. Kapitel 2.3.1). Da die verschiedenen Filter weiterhin für die gesamte Eingabe verwendet werden, verringert sich hier der benötigte Speicher stark. Man spricht hier von Parameter Sharing. Dies führt auch dazu, dass die Filter in der Lage sind, gleiche Merkmale an unterschiedlichen Positionen im Bild wahrzunehmen. Sie sind daher äquivariant für Translationen (GOODFELLOW ET AL. 2016:334).

Ist die Faltung abgeschlossen, wird auf das Ergebnis der linearen Faltungsoperation auch hier eine nicht-lineare Aktivierungsfunktion, meistens ReLU, angewendet (siehe Aktivierungsfunktionen). Häufig sind diese direkt nach jeder Faltungsschicht. Bei tieferen Netzen werden häufig mehrere Faltungen hintereinander vor der nächsten Aktivierungsschicht ausgeführt (vgl. ResNet50).

Neben der Faltung mit Aktivierung wird das Backbone eines CNNs noch von zwei weiteren Schichten beeinflusst: Den Pooling- und Batch Normalisierungsschichten (DERU & NDIAYE, 2019:91). Die Normalisierung des Batchs ist eine essenzielle Methode, um das Netzwerk zur schnelleren Konvergenz und besserer Generalisierung bei niedrigerer Trainingszeit zu bringen (OGUNDOKUN ET AL. 2022). Zu Beginn eines jeden Trainingsprozesses steht die Vorbereitung der Daten. Hierbei werden die Daten normalisiert, um zu gewährleisten, dass die Daten im gleichen Wertebereich liegen, und somit den gleichen Einfluss im Training besitzen.

Gerade bei tieferen Netzen können zusätzliche Schichten zu stärkeren Abweichungen der Wertebereiche führen. Deshalb wird klassischerweise nach jeder Faltung normalisiert, um das Training zu beschleunigen und den Einfluss vorheriger Layer aufzulösen (GARBIN ET AL. 2019). Formel 21 zeigt die mathematische Repräsentation der Batch Normalisierung (IOFFE UND SZEGEDY, 2015):

$$BN(x) = \gamma \frac{x_i - \mu_B}{\sqrt{\sigma_B} + \varepsilon} + \beta \tag{21}$$

Die eingehenden Instanzen des Minibatchs x_i werden mit dem Mittelwert μ_B des Batchs subtrahiert und durch die Wurzel der Varianz σ_B geteilt. Im Nenner wird noch der Parameter ε addiert, um eine Division durch Null zu vermeiden. Zusätzlich werden noch die Skalierung γ und die Verschiebung β als lernbare Parameter zur akkuraten Normalisierung jedes Batchs hinzugefügt.

Sind die Ausgaben der Faltungsschicht normalisiert, werden diese häufig in eine Pooling Schicht weitergeleitet. Das Pooling funktioniert hier ähnlich wie die Faltung. Es wird ein Filter über die Eingabe geschoben, um die markantesten Bereiche zu identifizieren und zu verstärken (DERU & NDIAYE, 2019:93). Der Unterschied zur Faltung ist, dass dieser Filter keine Gewichtungen besitzt. Je nach Operation wird bspw. der Maximalwert (Max Pooling) oder der Durchschnitt (Average Pooling) der Werte innerhalb des Filters übernommen.

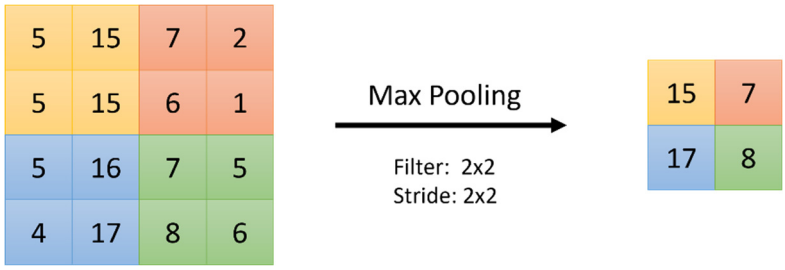


Abbildung 6: Beispiel einer MaxPooling Operation. Ein 4x4 Raster wird mit 2x2 Filter bei einer Schrittweite (Stride) 2 bearbeitet. Maximale Werte pro Iterationsschritt werden in Ausgaberaster übernommen (Eigene Darstellung).

Abbildung 6 verdeutlicht die Funktionsweise des Poolings anhand einer Max Pooling Operation. Es wird ein 2x2 Filter mit einer Schrittweite von 2 über das Eingangsbild geschoben, wodurch die Dimensionen reduziert und die markantesten Stellen im Bild hervorgehoben werden. Eine solche Operation eignet sich besonders in tieferen Netzen, wo Objekte mit niedrigem semantischem Wert und hoher räumlicher Auflösung mit Objekten mit hohem semantischem Wert und niedriger räumliche Auflösung kombiniert werden (vgl. Kapitel Semantische Segmentierung). Weiterhin wird durch Pooling die Translationsinvarianz gewährleistet, d.h. kleine Veränderungen im Bild wirken sich kaum bis gar nicht auf das Training aus (HOESER UND KUENZER, 2020).

Am Ende des Backbones werden die mehrdimensionalen Feature Maps in einen eindimensionalen Vektor zur Klassifikation im „Classifier Head“ überführt („Flattening“). Die extrahierten Features aus dem Convolutional Backbone können so mittels flachem und voll verbundenen KNN den definierten Ausgabeklassen zugewiesen werden. Mittels

Softmax Aktivierung wird eine Wahrscheinlichkeitsverteilung über alle Klassen ausgegeben (HOESER UND KUENZER, 2020).

Zum Abschluss dieses Kapitels soll noch auf zwei weit verbreitete Techniken im Kontext der Anwendung von CNNs eingegangen werden, da beides ebenfalls bei der Prozessierung im Rahmen dieser Arbeit verwendet wird: Data Augmentation und Transfer Learning. Beide Techniken sollen mit unterschiedlichen Ansätzen dem Overfitting während des Trainings vorbeugen. CNNs sind je nach Klassifizierungsaufgabe auf unzählige Daten angewiesen. Gerade bei Mehrklassenklassifikationen im Rahmen der Semantischen Segmentierung müssen Daten erhoben oder recherchiert werden, die alle Klassen in adäquater Weise abbilden können. Oft ist dies sehr zeit- und kostenintensiv. Eine Möglichkeit künstlich Daten zu generieren, ist Data Augmentation. Hierbei werden vorhandene Daten bspw. geometrisch verändert, mit Hilfe von Rotationen, Translationen oder Skalierungen. Neben der geometrischen Variante gibt es noch zahlreiche weitere Augmentierungen, die sich mit der Veränderung der Bildkanäle, oder der photometrischen Transformation auseinandersetzen. Dies sind zum Beispiel Helligkeitsveränderungen oder die vorherige Bearbeitung durch Filter wie Schärfen oder Weichzeichnen. Eine detaillierte Übersicht über die Möglichkeiten findet sich in SHORTEN UND KHOSHGOFTAAR (2019).

Neben der künstlichen Generierung von Trainingsdaten lässt sich auch auf vortrainierte Modelle zurückgreifen. Beim Transfer Learning werden die trainierten Gewichte eines anderen Netzes als initiale Gewichte für das eigene Netz verwendet. Das ist sehr effektiv zum Erlernen genereller Bildmerkmale, wie Kanten und Formen oder Änderungen der geometrischen oder photometrischen Eigenschaften. Erfahrungsgemäß lassen sich diese besser aus großen Datensätzen lernen (SHORTEN UND KHOSHGOFTAAR,2019). Auf der Basis dieser Gewichte werden nun die tieferen Schichten an die Besonderheiten der variierenden Eingangsdaten zur Lösung der neuen Klassifikationsaufgabe angepasst (GOODFELLOW ET AL. 2016:534). Ein sehr bekanntes Beispiel für Transfer Learning ist die ImageNet Challenge, wo unzählige Daten (1000 Klassen und über 1 Mio. Bilder) gesammelt und annotiert, und verschiedene DL-Modelle trainiert wurden. Die Gewichte dieser vortrainierten Modelle werden häufig als initiale Gewichte für ähnliche Klassifizierungsaufgaben verwendet (DENG ET AL. 2009).

2.3.3 Semantische Segmentierung

Die Semantische Segmentierung ist ein Verfahren, dass sich mit der Klassifizierung von Pixeln in einem Bild befasst. Hierbei wird jedem Pixel eine Klasse aus vorher definierten Klassen zugewiesen. Hierdurch wird dem Algorithmus die Klassenzugehörigkeit der einzelnen Pixel vermittelt. Als Ergebnis ergibt sich ein segmentiertes Bild, eingeteilt in die vordefinierten semantischen Klassen (HAO ET AL. 2020). Bevor tiefer in die Semantische Segmentierung eingestiegen wird, ist es wichtig diese von anderen gängigen Erkennungs- bzw. Klassifizierungsverfahren, innerhalb des maschinellen Sehens zu unterscheiden. Abbildung 7 gibt einen Überblick über diese verschiedenen Ansätze. Im Wesentlichen kann hier zwischen Objekterkennung und Segmentierung unterschieden werden. Bei der Bildklassifikation (engl. Image Classification (Abb. 7 (c))) handelt es sich um die Zuweisung von Labels für Objekte, die sich im Bild befinden, ohne Ausweisung der räumlichen Lage im Bild. Diese räumliche Lage wird bei der Objekterkennung (engl. Object detection (Abb. 7 (d))) in Form von sog. Bounding Boxes mit unterschiedlichen Klassenlabels ermöglicht. Diese Labels sind hier für Personen (rot) und Autos (blau) unterschieden. Eine Kombination aus Segmentierung und Objekterkennung bildet die Instanzsegmentierung (engl. Instance Segmentation (Abb. 7 (e))). Als Ergebnis erhält man ein segmentiertes Bild, in dem die Instanzen der vordefinierten Klassen kantenscharf voneinander unterscheidbar sind. Abb. 7 (e) zeigt dieses Vorgehen für Autos und Personen, während der Rest des Bildes NoData Werte sind.

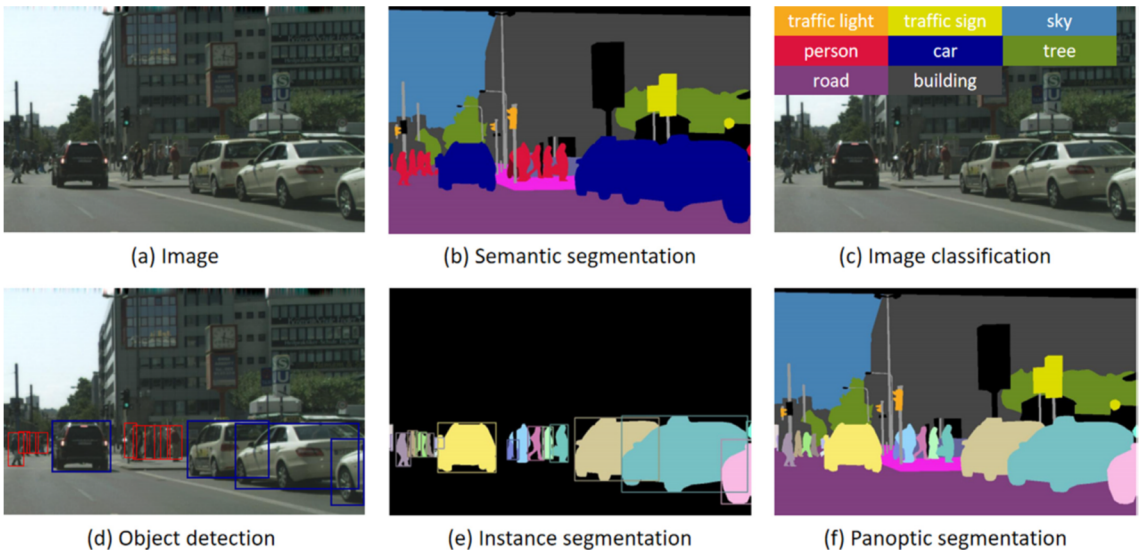


Abbildung 7: Überblick verschiedener Bildklassifizierungsverfahren. Dargestellt sind neben dem Ausgangsbild, Segmentierungsverfahren, Bildklassifikation (c) und Objektdetektion (d). Die Segmentierungsverfahren sind unterteilt in Semantische Segmentierung (b), Instanzsegmentierung (e) und panoptische Segmentierung (f) (HAO ET AL. 2020, KIRILLOV ET AL. 2019).

Verglichen mit der Semantischen Segmentierung (Abb. 7 (b)), wo lediglich Bereiche für die einzelnen Klassen ausgewiesen werden, bietet die Instanzsegmentierung in Bezug auf die Abgrenzung der einzelnen Instanzen innerhalb der Klasse einen großen Mehrwert. Die panoptische Segmentierung (Abb. 7 (f)) bringt diesen Ansatz noch einen Schritt weiter, indem jedem Pixel ein semantisches Label und ein Instanzlabel zugewiesen werden soll (HAO ET AL. 2020; KIRILLOV ET AL. 2019). Im Rahmen dieser Arbeit wird der Ansatz der semantischen Segmentierung verfolgt. Für die Ausweisung von Landbedeckungsklassen reicht es aus zwischen den Klassen und nicht den Klasseninstanzen zu differenzieren. Die Semantische Segmentierung blickt auf eine lange und umfangreiche Publikationshistorie zurück und ist schon lange ein essenzieller Bestandteil des maschinellen Sehens (ZHU ET AL. 2016). Aktuell existierende methodische Ansätze können je nach Überwachungsgrad in drei Hauptklassen eingeteilt werden: Unüberwachte, wenig/halb überwachte und voll überwachte Ansätze. Hierbei sei angemerkt, dass eine klare Abgrenzung zwischen diesen Methoden schwierig ist. Unüberwachte Ansätze beschäftigen sich mit der Klassifikation auf Basis homogener Pixelwerte ohne Zuweisung vorheriger Labels (vgl. Kapitel 2.2). Da die Erstellung gelabelter Trainingsdaten sehr zeitaufwendig ist und je nach Klassifikationsaufgabe keine geeigneten Daten zur Verfügung stehen, ist es wichtig, die Segmentierung auf Daten mit wenig Labels zu untersuchen. Solche wenig überwachten (engl. Weakly- / Semi-supervised) Ansätze haben mit kaum annotierten Daten hohe Klassifizierungsgenauigkeiten zu erzielen (HAO ET AL. 2020). Liegen diese Annotationen für die gesamten Trainingsdaten vor, spricht man voll überwachten Ansätzen. Neben klassischen Verfahren (z.B. Random Forst (BREIMAN, 2001) und Multi-Class Support Vector Machine (WANG UND XUE, 2014)) erleben DL-Ansätze wachsende Beliebtheit im Bereich der vollüberwachten Segmentierung. Eine gängige Netzarchitektur sind hier die Fully Convolutional Networks (FCNs) (LONG ET AL. 2015). FCNs sind sog. Autoencoder oder Encoder-Decoder Systeme. Ziel dieser Systeme ist es eine Synergie zwischen Outputs von tiefen und flachen Schichten in einem Netz zu schaffen. Der Encoder komprimiert die Eingangsdaten durch Faltungs- und Poolingoperationen, wodurch der semantische Wert mit jeder weiteren Schicht auf Kosten räumlicher Details zunimmt. Am Ende des Encoders folgt der Decoder, welcher die Eingangsdaten dekomprimiert und versucht wiederherzustellen. Das Ergebnis wird im Anschluss mit dem erwarteten Wert verglichen. FCNs besitzen zur Performancesteigerung, sog. „Skip Connections“. Hierbei werden Feature Maps der mittleren Schichten des Encoders an den Decoder übergeben, um

diesen im „Upsampling“ der Daten zu unterstützen (BADRINARAYANAN ET AL. 2017; HAO ET AL.2020). Dadurch ist es möglich sehr fein aufgelöste Segmentierungen mit gesteigerter Genauigkeit zu generieren (YU ET AL. 2018). Eine bekannte Gruppe von Autoencodern sind die „U-Nets“ (RONNEBERGER ET AL. 2015). Abb. 8 zeigt den schematischen Aufbau eines U-Net als Repräsentant der Autoencoder.

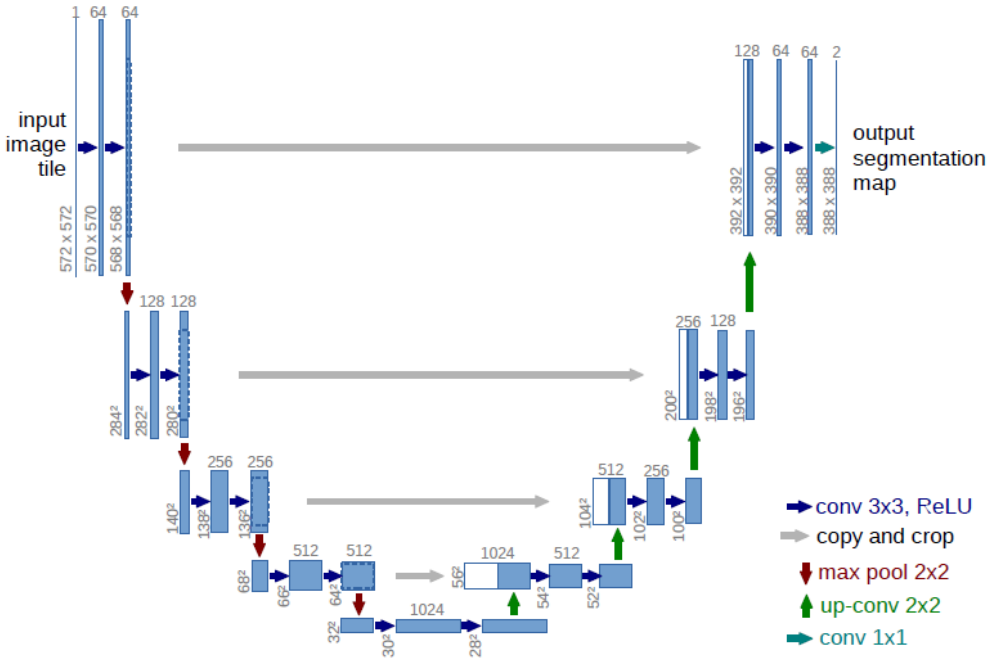


Abbildung 8: Modellarchitektur eines U-Nets. Dargestellt ist die schematische Verarbeitung der Eingangsdaten. Diese werden in einer bestimmten Abfolge an Schichten im Encoder (links) gefaltet und anschließend wird im Decoder ein Upsampling durch Beteiligung der skip sonnections (grau) vorgenommen (RONNEBERGER ET AL. 2015).

Die linke Seite zeigt den Encoder und dessen Faltungen der Eingangsdaten. Auf der rechten Seite befindet sich der Decoder mit dessen Upsampling der Ausgabe des Encoders. In grau sind die „Skip Connections“ zu sehen. Beim Upsampling im Decoder lässt sich an den Dimensionen der finalen Segmentation Map erkennen, dass diese nicht die Auflösung der Eingangsdaten erhält. Weiteres Upsampling bringt hier hohe Speicherlast und in der Regel wenig bis keine Performancesteigerung (BADRINARAYANAN ET AL. 2017). Das U-Net und dessen Popularität ist ein Beispiel für die Effizienz und Genauigkeit dieser Architekturen für Segmentierungsaufgaben. Es haben sich im vergangenen Jahrzehnt zahlreiche Architekturen entwickelt (z.B. RefineNet (LIN ET AL 2017), SegNet (BADRINARAYANAN ET AL. 2017) oder das FC-DenseNet (JEGOU ET AL. 2017)). Hierbei werden unterschiedliche Konfigurationen des Encoders und des Decoders gewählt. Im Rahmen

dieser Arbeit wird ein ResNet50 als Encoder und ein Feature Pyramid Network (FPN) als Decoder verwendet. Diese werden in den folgenden Kapiteln detaillierter dargestellt.

2.3.4 Residual Networks (ResNet)

Die Bekanntheit des ResNets von HE ET AL. (2015) ist auf dessen Sieg bei der ImageNet Challenge im Jahr 2015 zurückzuführen. Der Erfolg des ResNets liegt in dessen Architektur. Wie in vorherigen Kapiteln erwähnt, kommt es im Rahmen der Backpropagation häufig zu sehr kleinen Gradienten, was sich negativ auf die Aktualisierung der Gewichte auswirkt (vgl. Kapitel Aktivierungsfunktionen). Durch Einführung der sog. „Residual Units" aus denen ResNets aufgebaut sind, soll diese Problematik minimiert werden. Residual Units zeichnen sich durch „Skip connections" aus. Hier wird dem Ergebnis aus jedem Faltungsblock das Ergebnis der vorherigen Schicht hinzugefügt. Diese Skip Connection ermöglicht eine stabilere Backpropagation, da somit eine direkte Verbindung vom Klassifizierungslayer in die oberen Schichten des Netzwerkes gibt.

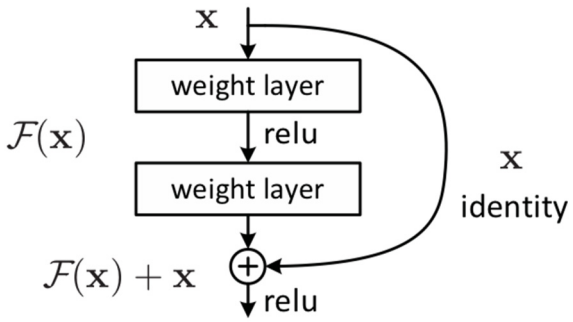


Abbildung 9: Schematische Darstellung einer Skip Connection am Beispiel eines ResNets. Innerhalb solcher Residual Units wird jedem Faltungsblock (F(x)) das Ergebnis der vorherigen Schicht (x) hinzugefügt (HE ET AL.2015).

Abbildung 9 zeigt die Funktionsweise einer Skip Connection im Rahmen eines sog. Identity Blocks. Das Ergebnis aus der vorherigen Schicht x wird hier als Identity bezeichnet und dem Ergebnis der Faltung F(x) hinzugefügt. Beides läuft in die Aktivierung (hier ReLU) und wird zur nächsten Schicht weitergeleitet. Die Schichtenfolge innerhalb dieser Identity Blocks variiert mit der Entwicklung von immer tieferen Netzen, die durch die Einführung der ResNets möglich waren. Bekannte Netzarchitekturen der ResNets sind in Tabelle 2 dargestellt. HE ET AL. (2015) evaluierten in dieser Ausarbeitung verschiedene ResNet Architekturen in Rechenzeit und Genauigkeit. Trainiert wurden die Modelle auf dem ImageNet 2012 Klassifikationsdatensatz, mit 1.28 Mio. Trainingsbildern und 50k

Validierungsbildern. Die Unterscheidung der ResNets richtet sich nach der Anzahl der Schichten, demnach besitzt ein ResNet50 50 Schichten, aufgeteilt in:

- Ein Convolutional Layer mit 7x7 Kernel, der Schrittweite von 2 und 64 Filtern als Ausgabe
- Ein MaxPooling Layer (3x3, Schrittweite 2)
- Der zweite Faltungsblock (conv2_x) besteht aus 3x3 Schichten mit einer 1x1 (64 Filter), einer 3x3 (64 Filter) und einer 1x1 (256 Filter) Faltung.
- Im Anschluss folgt der dritte Faltungsblock (Conv3_x) mit 4x3 Schichten ((1x1, 128 Filter), (3x3, 128 Filter) und (1x1, 512 Filter))
- Daraufhin Con4_x mit 6x3 Schichten ((1x1, 256 Filter), (3x3, 256 Filter) und (1x1, 1024 Filter))
- Der finale Faltungsblock schließt mit 3x3 Schichten ((1x1, 512 Filter), (3x3, 512 Filter) und (1x1, 2048 Filter)) die Faltungen im ResNet50 ab.
- Abschließend erfolgt ein Average Pooling und ein „Flattening“ mit anschließender Einspeisung in ein voll vernetztes Netzwerk, dessen Ausgabe eine Wahrscheinlichkeitsverteilung über die Klassen ist (aufgrund von Softmax).

Die Spalte Output Size verdeutlicht, dass in den Faltungsschichten eine Dimensionsreduktion (Faltung mit Schrittweite 2) durchgeführt wird. Aus diesem Grund sind neben den Identity Blocks ebenfalls Convolutional Blocks zu verwenden. Diese passen die Dimensionen der Skip Connections von der vorherigen Schicht auf die aktuelle Schicht an (HE ET AL. 2016). Da diese Dimensionsreduktionen am Anfang eines Faltungsblocks durchgeführt werden, ist der Convolutional Block vor nachfolgenden Identity Blocks einzubauen. Durch die Einführung der Residual Units und ihrer Skip Connections und deren positive Auswirkung auf die Backpropagation ließen sich immer tiefere Netzarchitekturen gestalten. HE ET AL. (2015) zeigen, dass selbst Netze mit 152 Layern (ResNet152) noch gut generalisieren konnten und zu einer Minimierung des Verlusts führten. Die Erweiterung der Netztiefe führt allerdings nicht unbegrenzt zur Verbesserung der Modellperformance. So stieg der Verlust zwischen ResNet110 auf ResNet1202 von 6.43% auf 7.93% an (HE ET AL. 2015). Eine Umgestaltung der Residual Units kann zur Performancesteigerung bei sehr tiefen Netzen führen. HE ET AL. 2016 zeigten hier verschiedene Möglichkeiten, die sich auf unterschiedliche Schichtenabfolgen und deren Einfluss auf die Performance bezogen.

Tabelle 2: Übersicht verschiedener ResNet Architekturen mit variierender Anzahl an Schichten. Diese variieren innerhalb der Faltungsblöcke (conv2_x bis conv5_x). Conv1 ist bei allen Architekturen identisch. Für die einzelnen Faltungen ist im Filtergröße (z.B. 1x1) und Anzahl der Filter (z.B. 64) notiert. Zusätzlich dargestellt ist sinkende Bildgröße (output size) bei Durchlauf der Faltungsschichten. Die untere Zeile stellt zudem die wachsenden FLOPS bei steigender Anzahl an Schichten dar (HE ET AL. 2015).

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

Nutzt man das ResNet als „convolutional backbone“ zur Extraktion vorhandener Features, sollte man diesen unter Berücksichtigung der Inputdaten gestalten. Hier spielen die Tiefe, die Anzahl der zu extrahierenden Klassen sowie deren Repräsentation und Rechenlast eine entscheidende Rolle. Vor dem Hintergrund, dass der Ursprung des ResNets im Bereich des Maschinellen Sehens liegt, wo eine enorme Menge an unterschiedlichen Klassen erkannt werden müssen, ist dies bei klassischen Landbedeckungsklassifikationen oft nicht der Fall. Daher liefern die tiefsten ResNets wie bspw. das ResNet-152 (bestehend aus 152 Layern), aufgrund der geringeren Anzahl an zu erkennenden Klassen in Fernerkundungsdaten, oft schlechtere Genauigkeiten als flachere Netze (HOESER ET AL. 2020). Um Overfitting zu vermeiden, tendiert man somit eher zu flacheren Netzen. Das Modeldesign muss hier der gewünschten Aufgabe inkl. Daten angepasst werden.

2.3.5 Feature Pyramid Networks (FPN)

Die Nutzung von Bildpyramiden zur Erkennung von Objekten in unterschiedlichen Maßstäben ist bereits seit langem ein fester Bestandteil des maschinellen Sehens. Dies wird durch bekannte Algorithmen zur Merkmalsextraktion aus Bildern, wie SIFT („Scale-invariant feature transform“, LOWE, 2004) und HOG („Histogram of oriented Gradients“, DALAL UND TRIGGS, 2005), bestätigt. Die Stärke in der Verwendung von Bildpyramiden ist die Merkmalsextraktion auf verschiedenen Bildmaßstäben. Die Bildung sog. „Feature

Image Pyramids“ hat ein einen großen Mehrwert für die Objekterkennung und Segmentierung, da die hochaufgelösten Schichten vom höheren semantischen Wert der tieferen Schichten profitieren (LIN ET AL. 2017). Die Berechnung solcher „Feature Image Pyramids“ ist allerdings häufig sehr speicher- und zeitintensiv. Durch die Entwicklung der CNNs erhielt man hier architekturbedingt einen entschiedenen Vorteil. Wie in den vergangenen Kapiteln geschildert, geben CNNs Feature Maps aus. Mit zunehmender Netztiefe nimmt die räumliche Auflösung dieser Maps ab, gewinnt aber an semantischem Wert. Diese Fähigkeit macht man sich zu Nutze und greift die Feature Maps der verschiedenen Schichten ab und sendet sie, via lateraler Verbindung, an das FPN. Man nutzt demnach die hierarchische Feature-Pyramide des CNNs und spart sich dadurch Zeit und Speicher.

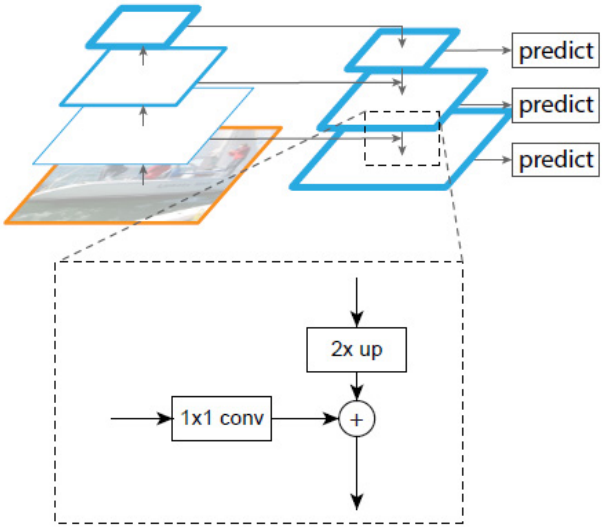


Abbildung 10: Aufbau eines Feature Pyramid Networks (FPN). Zu sehen sind der Convolutional Backbone im Bottom-Up-Pathway auf der linken Seite sowie das FPN im Top-Down-Pathway der rechten Seite. Der untere Bereich des Bildes zeigt zudem die Verarbeitung der eingehenden Feature Maps via lateraler Verbindungen (LIN ET AL. 2017)

Zur Erstellung der Feature Pyramide werden ein sog. „Bottom-Up-Pathway“ und ein „Top-Down-Pathway“ verwendet. Abbildung 10 zeigt diesen Aufbau. Als Bottom-Up-Pathway wird in der Regel der Convolutional Backbone (linke Pyramide, Abb. 10) bezeichnet, in der die Feature Maps gebildet werden. Zusätzlich erfolgt hier pro Faltungsschicht eine Dimensionsreduktion um den Faktor 2. Das Ergebnis des letzten Faltungsblocks aus dem Backbone, leitet den Top-Down-Pathway (rechte Pyramide, Abb. 10) und somit den Beginn des Feature-Pyramidenaufbaus ein. Nach Lin et al. 2017 wird hier eine 1x1 Convolution (256 Filter) verwendet (dasselbe gilt für die lateralen Verbindungen zwischen Bottom-Up und Top-Down

Pathway). Nach der initialen Faltung kommt es zum Upsampling um den Faktor 2 unter Verwendung der nächsten-Nachbarn. Die entstandene Feature Map wird mit den Feature Maps gleicher räumlicher Auflösung aus dem Bottom-Up-Pathway zusammengeführt, nachdem diese ebenfalls eine 1x1 Faltung mit 256 Filtern durchlief (Dimensionsangleichung). Durch diese Zusammenführung macht man sich die präziseren räumlichen Eigenschaften der Aktvierungen (Bottom-Up) beim Upsampling zu Nutze. Im Anschluss erfolgt die nächste Iteration, bis alle Feature Maps erstellt sind. Jede zusammengeführte Map durchläuft noch eine 3x3 Convolution um dem Aliasing Effekt durch das Upsampling zu minimieren. Da alle entstandenen Feature Maps am Ende noch zusammengeführt werden und mit gleichen Klassifikatoren untersucht werden, ist die Anpassung der Dimension (256 Kanäle) in den lateralen Verbindungen essenziell. Weiterhin sei noch angemerkt, dass typischerweise nicht bis zur feinsten Auflösung iteriert wird, da die Berechnung hier zu speicherintensiv ist (LIN ET AL. 2017). Im Falle der Verwendung des ResNets50 mit fünf Faltungsblöcken, würde hier lediglich bis Conv2 iteriert werden (vgl. Abb. 10).
Trotz der starken Performance der CNNs im Bereich der Featureextraktion und der Robustheit gegenüber skalenabhängigen Varianzen der Daten, kann die Genauigkeit unter Verwendung von FPNs gesteigert werden. Die erstellten Feature Image Pyramiden des Top-Down-Pathway tragen speziell bei der Rechnung der Inferenz zur Steigerung der Genauigkeit bei und werden daher in vielen Ansätzen zur Objekterkennung oder -segmentierung verwendet (z.B. SEFERBEKOV ET AL. 2018; WANG UND ZHONG, 2021, LIN ET AL. 2017).

2.4 Deep Learning Frameworks

Mit der wachsenden Popularität von KI-Ansätzen innerhalb des letzten Jahrzehnts wurden zahlreiche Frameworks entwickelt. Zu den bekanntesten Vertretern gehören hier Tensorflow mit Keras (ABADI ET AL. 2016, TENSORFLOW 2024, KERAS 2024, MATHEW ET AL. 2021), PyTorch mit Lightning (PYTORCH 2024), Microsoft Cognitive Toolkit (CNTK, CNTK 2024), Scikit-learn (SCIKIT-LEARN 2024), Caffe (CAFFE 2024) und Theano (DERU & NDIAYE 2019:45, MINAR & NAHER 2018):

- Tensorflow mit Keras: Die erste Version von Tensorflow stammt aus 2015 und wurde 2017 als Open Source vom Google Brain Team veröffentlicht. Der Name Tensorflow leitet sich hier aus der Handhabung der Daten ab. Tensoren sind mehrdimensionale Vektoren, die durch die neuronalen Netze (oder Graphen) fließen. Mittlerweile bietet das Framework zahlreiche Schnittstellen zu Python, C++, Java

oder JavaScript an. Zusätzlich steht seit Version 2.0 Keras als feste High Level API zur Verfügung. Dadurch wird dem Nutzer der Einstieg in die Arbeit mit neuronalen Netzen und der Zugang zu Tensorflow erleichtert. So können zahlreiche methodische Ansätze (z.B. die Arbeit mit CNNs oder RNNs) in verschiedenen Hardwarekonfiguration (z.B. Training auf CPUs und GPUs) getestet werden (MATHEW ET. AL 2018). Aufgrund ihrer Popularität haben Tensorflow und Keras eine breite Community mit schnellem Support, was die Nutzung zunehmend erleichtert (DERU & NDIAYE 2019:45).

- PyTorch mit Lightning: PyTorch wurde 2016 von der Facebook AI Research Group (FAIR) veröffentlicht. Hierbei handelt es sich um eine Python Library, die auf dem in 2002 veröffentlichten Framework Torch (in Skriptsprache Lua geschrieben) aufbaut. Neben Python steht zusätzlich eine C++ Schnittstelle zur Verfügung. Die Nutzung von PyTorch wird durch deren High-Level API „Lightning" erleichtert. Dadurch ist es eine leistungsstarke Option zur Tensorenberechnung und der Gestaltung von neuronalen Netzen. Aufgrund der höheren Flexibilität und Geschwindigkeit gegenüber Tensorflow erfreut sich PyTorch hoher Beliebtheit im wissenschaftlichen Kontext. Wie Tensorflow bietet PyTorch sowohl CPU als auch GPU-Unterstützung (PYTORCH 2024, MATHEW ET AL. 2018, DERU & NDIAYE 2019:45). In dieser Arbeit wurde sich aufgrund bereits vorhandener Kenntnisse und der gesteigerten Geschwindigkeit in der Berechnung für PyTorch mit dessen High-Level API „Lightning" entschieden.
- Caffe, CNTK und Theano: Caffe, CNTK und Theano sind Deep-Learning-Frameworks von Microsoft (CNTK, in 2016), dem „Vision and Learning Center der University of California" (Caffe, in 2014) und der Universität von Montreal (Theano, in 2007). Alle Frameworks bieten eine GPU-Unterstützung und unterschiedliche Schnittstellen an. Während Caffe und CNTK in C++ geschrieben wurden, basiert Theano auf Python. CNTK und Caffe bieten jedoch ebenfalls Python Schnittstellen an. Alle Frameworks können für die Arbeit mit unterschiedlichen Modellen (z.B. CNNs und RNNs) verwendet werden (CAFFE 2024, CNTK 2024, THEANO 2024, MATHEW ET AL. 2018, DERU & NDIAYE 2019:45).

2.5 Deep Learning in der Erdbeobachtung

Die Ableitung von Landbedeckungsinformationen hat bereits eine lange Historie im Bereich des maschinellen Lernens. Durch den rasanten Anstieg an Deep Learning Verfahren für die Detektion von Objekten oder der Segmentierung von Bildern gab es im letzten Jahrzehnt zahlreiche Publikationen, die sich mit der Ableitung der Landbedeckung befassten. Nach HOESER UND KUENZER (2020) nimmt die Generierung von Landbedeckungs- und Landnutzungskarten (LCLU) mit 13% der untersuchten Publikationen den dritten Platz ein. Darüber lagen nur Ansätze des Verkehrssektor (27%, z.B. die Detektion von Schiffen oder Flugzeugen) und der Siedlungsbereich (26%, z.B. Detektion von Gebäuden) (HOESER UND KUENZER, 2020). Von den 57 Publikationen des LCLU-Sektors verfolgten lediglich 27 Publikationen (von insgesamt 429) die Thematik der mehrklassigen pixelweisen Ableitung von Landbedeckungs- oder Landnutzungskarten wie es diese Arbeit macht. Dies gibt einen ersten Eindruck zum Ungleichgewicht zwischen Publikationen im Bereich der Objektdetektion und dem Bereich der Segmentierung.

Semantische Segmentierung erfährt speziell seit der Verwendung von DCNNs („Deep Convolutional Neural Networks“) von LONG ET AL. (2015) wachsendes Interesse in der Anwendung auf Fernerkundungsdaten. Einer der populärsten methodischen Ansätze sind die Encoder-Decoder Systeme (z.B. Wurm et al. 2019, Wurm et al. 2021). Im Gegensatz zur Anwendung im Bereich der Computer Vision birgt die Anwendung auf Erdbeobachtungsdaten einige Herausforderungen. Erdbeobachtungsdaten weisen eine geänderte Perspektive bei der Bildaufnahme auf, und sind im Gegensatz zur Computer Vision stets von oben aufgenommen. Speziell für vortrainierte Modelle kann dies herausfordernd sein, da Objekte nun anders wirken. Weiterhin liegen die Ursprünge der Computer Vision und dem damit verbundenen Einsatz von DL-Modellen bei RGB-Bildern auf Basis derselben Sensoren (HOESER UND KUENZER, 2020). Bei Erdbeobachtungsdaten variieren die Anzahl der Bildkanäle sowie die eingesetzten Sensoren je nach Erdbeobachtungsmission und eingesetzter Satelliten (SHERRAH, 2016). Weitere Herausforderungen in Erdbeobachtungsdaten sind die Auflösung der Eingangsdaten, die Lage der Objekte innerhalb des Bildes sowie die generelle Klassenverteilung der Objekte. Die Struktur der Klassen/Objekte ist verglichen mit natürlichen Bildaufnahmen wesentlich dichter in Erdbeobachtungsdaten (HOESER UND KUENZER, ZHANG ET AL. 2019). Betrachtet man den Umgang mit der Auflösung, kommt es in dieser Arbeit zur Anpassung von 5m (RapidEye) auf 3m (Pla-

netScope). Die Daten beider Missionen zählen zu den hochaufgelösten Erdbeobachtungsdaten (MA ET AL. 2019). Niedrigere räumliche Auflösung führen zu unscharfen Grenzen zwischen einzelnen Landbedeckungsklassen und bestärken die Mischpixelproblematik im Randbereich der Polygone. Nichtsdestotrotz hat die Einführung der skip connections zwischen Encoder und Decoder maßgeblich zur Glättung der Landbedeckungsgrenzen geführt (KEMKER ET AL. 2018). Gerade auch bei sehr hochaufgelösten Eingangsdaten (<1m GSD) hilft dies bei der Reduzierung sog. Salt-and-Pepper Fehler (PINHEIRO ET AL. 2016). Eine Erhöhung der Auflösung steigert die Komplexität der Objekte und führt zu zunehmender Heterogenität innerhalb der Objekte einer Klasse (WANG ET AL. 2017). Um sich diesen Problemen anzunehmen, bedienen sich unzählige Publikationen sog. Benchmark Datensätze. Klassische Vertreter sind hier die Vaihingen- und Potsdam-Datensätze der ISPRS („International Society for Photogrammetry and Remote Sensing“) (ROTTENSTEINER ET AL. 2012). Mit einer GSD von 5-9cm weisen diese eine sehr hohe räumliche Auflösung auf, sind allerdings räumlich auf die genannten Städte begrenzt. Zusätzlich wird lediglich zwischen fünf Landbedeckungsklassen unterschieden. Im Laufe des letzten Jahrzehnts entwickelten sich weitere Datensätze mit ähnlicher räumlicher Auflösung aber steigender Klassenvielfalt, z.B. EvLab-SS (ZHANG ET AL. 2017) mit 10 Klassen oder RIT-18 (KEMKER ET AL.2018) mit 18 Klassen. Hierbei ist EvLab-SS mit WorldView-2 eine der wenigen satellitengestützten Missionen, häufig werden regional begrenzte Luftbilder oder UAS („Unmanned Airborne System“, z.B. Dronen) Daten verwendet (HOESER UND KUENZER, 2020; KEMKER ET AL. 2018). Trotz der Menge an frei verfügbaren Satellitenbildern in mittlerer Auflösung (z.B. Landsat und Sentinel (10-30m)) sind Landbedeckungssegmentierungen schwer möglich. Hier fehlt den Daten die nötige räumliche Auflösung zur Darstellung feiner Objektstrukturen (SHARMA ET AL. 2017). Weiterhin sind offene Datensätze häufig auf spezielle Domänen konzentriert, wie die Erkennung und Segmentierung von Gebäuden (SPACENET BUILDING 2024) oder Straßennetzen (MNIH 2013; HOESER ET AL. 2020). Wenig davon befasst sich mit mehrklassigen Segmentierungen, geschweige denn liegen diese flächendeckend vor. Mit dem LBM-DE und dessen bundesweite Ausprägung liegt dieser Arbeit ein Datensatz mit viel Potential vor. Somit können je nach Verfügbarkeit von luft- oder satellitengestützten Befliegungsdaten, variable Region der Bundesfläche über Deutschland mit hochwertigen Labels versehen werden. Ein vergleichbarer Ansatz wird in dieser Arbeit für das Bundesland Schleswig-Holstein unter Verwendung der PlanetScope Daten gemacht. Die Menge an Eingangsdaten

sowie die Qualität der Datenlabels bestimmt hier maßgeblich die Güte des Modells. Weniger Eingangsdaten bergen das Risiko von Overfitting bzw. sehr komplexen Modellen, um die benötigten Informationen aus den Daten zu beziehen (LATEEF UND RUICHEK, 2019). Gerade bei wenigen Eingangsdaten, wie der Abdeckung eines Bundeslandes sind genaue Labels entscheidend. Sind selten vorkommende Klassen nun unzureichend gelabelt, ist das Modell nicht in der Lage diese zu erfassen. Eine Kontrolle der Labels, speziell der Klassen, die nicht während jeder Aktualisierungsperiode angeschaut werden, würde sicherlich auch im Falle des LBMs zur Steigerung der Genauigkeiten führen (Kapitel 4). Nichtsdestotrotz ist die Klassentiefe mit 31 Klassen in dieser Arbeit maßgeblich zur vorangegangenen Literatur erhöht und deshalb auch nur bedingt vergleichbar. Dennoch sollen nun einige Landbedeckungssegmentierungen folgen, um die Ergebnisse dieser Arbeit besser einschätzen zu können. Das erste Beispiel ist die „DeepGlobe 2018 Challenge Satellite Image Understanding Challenge“ (DEMIR ET AL. 2018). Hier wurden sieben Landbedeckungsklassen (Urban, Agriculture, Rangeland, Forest, Water, Barren und Unknown) mit einer Auflösung von 50cm/Pixel (Herkunft: DigitalGlobe + Vivid) untersucht. Hierbei umfasste der Datensatz in Gänze ca. 10K Satellitenbilder und die erreichte mIoU beträgt 0.433. Diese wurde maßgeblich durch kleinere und komplexere Objektstrukturen gemindert, wohingegen größere einfachere Objektstrukturen besser funktionierten (DEMIR ET AL. 2018). Weitere Herausforderungen sind große Variabilität innerhalb einzelner Klassen mit wenig Unterschieden zu anderen Klassen (HUANG ET AL. 2020). Beispielsweise bestehen große Unterschiede zwischen bewirtschafteten und nicht bewirtschafteten Agrarflächen, wohingegen Agrarflächen je nach Bewuchs starke Ähnlichkeiten zu Grünlandflächen haben können.

Als nächstes Beispiel dient die DIResUNet Architektur von PRIYANKA ET AL. (2022). Die Modellarchitektur besteht aus einem Interception Modul, einem modifizierten residual Block, einem sog. „dense global spatial pyramid pooling (DGSPP)“ und dem U-Net Schema (näheres hierzu in PRIYANKA ET AL. 2022). Dieser Ansatz wurde nun auf Basis des WHDLD („Wuhan dense labelling dataset“) Datensatzes (2m/px) (SHAO ET AL. 2018) aus satellitengestützten Missionen mit anderen Architekturen verglichen. Zusätzlich wurden noch Luftbilder aus dem Landcover.ai Datensatz (BOGUSZEWSKI ET AL. 2020) mit einer räumlichen Auflösung von 50cm/px evaluiert. Unterschieden wurde hier bei Landcover.ai in drei (Water, Building, Woodland) und bei WHDLD in sechs Landbedeckungs-

klassen (Water, Vegetation, Bare Soil, Pavement, Building, Road). So erzielte DI-ResUNet für Wasser und Vegetation/Baumbestand IoU-Werte von 40,26% (Wasser) und 60,68% (Vegetation). Die IoU Werte hängen oftmals eng mit der Beschaffenheit der Eingangsdaten zusammen. Der WHDLD Datensatz deckt urbane Regionen Wuhans ab. Demnach fließen viele Trainingsdaten aus dem Bereich Bebauung (hier als Building mit einer IoU von ca. 80%) und weniger Trainingsdaten der anderen LB-Klassen ein (PRIYANKA ET AL., 2022). Diese Beispiele geben einen guten Einblick in eine weitere Herausforderung in der Anwendung neuronaler Netze auf Fernerkundungsdaten – dem Klassenungleichgewicht (HUANG ET AL. 2020). Hierbei sind speziell Klassen mit wenig Vorkommen oder sehr vereinzelter räumlicher Verbreitung benachteiligt. Das Resultat sind wenige Pixel pro Klasse, was in stark unzureichender Klassifizierungsgüte für die jeweilige Klasse endet (HUANG ET AL. 2020). Solche Klassen würden durch eine höhere Auflösung, sowie räumlich differenzierterer Labels profitieren, sodass Vegetation von urbanen Strukturen abgegrenzt werden kann. Ein solches Anpassen von Labeln ist allerdings kostspielig und sehr zeitintensiv, hebt jedoch erneut die Bedeutsamkeit qualitativ hochwertiger Labels hervor (HUANG ET AL. 2020).

Die Postprozessierung von Ergebnissen ist nach MA ET AL. (2019) eine der vier Strategien, denen man sich in der Semantischen Segmentierung bedient, um die Genauigkeiten zu steigern und die Hürde zwischen räumlicher Auflösung und semantischem Wert zu reduzieren. Im letzten Jahrzehnt gab es noch weitere Ansätze, die mit der Umgestaltung von Netzarchitekturen (z.B. Atrous Convolution im Encoder Netzwerk oder die Hinzunahme von Objekten unterschiedlicher Auflösung (SHERRAH ET AL. 2016, MARMAMIS ET AL. 2016)). Weiterhin wurden die Möglichkeiten von nicht gefalteten Schichten und Skip Connections im bzw. zum Decoder Netzwerk (KEMKER ET AL. 2018) sowie der Einfluss von der Kombination von mehreren Netzen mit unterschiedlichen Initialisierungen getestet (MARMAMIS ET AL. 2016; MA ET AL. 2019). Als leistungsfähige Option für mehrklassige semantische Segmentierung haben sich die Encoder-Decoder Systeme mit der ResNet Familie als Convolutional Backbone erwiesen (HOESER UND KUENZER, 2020). Hierbei trägt der Übertrag von Feature Maps in unterschiedlicher Auflösung maßgeblich zur Steigerung des Klassifizierungsergebnis der Segmentierung bei (CHEN ET AL. 2017, HOESER UND KUENZER, 2020).

3 Methodik und Eingangsdaten

Im folgenden Kapitel erfolgt ein detaillierter Überblick über die Eingangsdaten des Modelltrainings sowie der strukturierte Aufbau des verwendeten Modells. Anschließend wird die verwendete Software skizziert. Initial wurde die Modellarchitektur mit einer deutschlandweiten Abdeckung von RapidEye aus dem Jahr 2018 trainiert. Hierbei diente das LBM 2018 als Maskierung und lieferte somit die passenden Landbedeckungsklassen zu den eingesetzten Satellitenszenen. Dieses vortrainierte Modell wurde im Rahmen dieser Arbeit auf Planet Labs Daten übertragen und für den Domänenwechsel angepasst. Mittels Transfer Learning wurde das bestehende Modell nachtrainiert und somit an die Planet Labs Daten aus 2021 angepasst. Hierbei wurden LBM 2021 Daten als Maskierungslayer für die Semantische Segmentierung für Schleswig-Holstein eingesetzt. In den folgenden Abschnitten werden RapidEye (3.1), Planet Labs (3.2), das LBM (3.3) und die Modellarchitektur (3.4) erläutert. Im Anschluss folgen die Vorbereitung der Daten (3.5), der Trainingsprozess und die Hyperparameteroptimierung (3.6) sowie die verwendete Software (3.7).

3.1 RapidEye

Die RapidEye Mission wurde im Jahr 2008 mit dem Launch des ersten der insgesamt fünf Satelliten ins Leben gerufen. Ursprünglich wurden die Satelliten vom privaten deutschen Unternehmen RapidEye AG betrieben, gingen aber, nach dessen Insolvenz im Jahr 2011, in den Besitz des amerikanischen Konzerns Planet Labs über (DLR 2023). Ende März 2020 wurde der gesamte Betrieb von RapidEye von Planet Labs eingestellt. Alle Satelliten waren mit optischen Kameras ausgestattet und konnten, je nach gewünschtem Produkt, bis zu fünf spektrale Kanäle zur Verfügung stellen. Neben den **R** (Rot) **G** (Grün) **B** (Blau) Kanälen tragen **RE** (RedEdge) und **NIR** (Nahes Infrarot) zu einer multispektralen Abdeckung von 440nm bis 850nm bei. Die fünf Satelliten umkreisten die Erde in einer sonnensynchronen Umlaufbahn in ca. 630km Höhe. Mit einer Schwadbreite von ca. 80km und einer maximalen Bildstreifenlänge von 1500km konnten pro Tag großflächige Gebiete der Erdoberfläche aufgenommen werden. Die geometrische Auflösung betrug hierbei bis zu 5m/Pixel (PLANET LABS 2023(A)). Aktuell vertreibt Planet Labs verschiedene Produkte in unterschiedlichen Prozessierungsstufen. Neben Szenen (75x50km² bis 75x300km²) in Prozessierungsstufe L1B können ebenfalls sog. Orthotiles (25x25km²) in

Stufe L3A erworben werden. Die Prozessierungsstufen umfassen radiometrische und Sensorkorrekturen (L1B) bis hin zur Orthorektifizierung mittels RCPs (Rational Polynomial Coefficients) und Höhenmodell (L3A) (PLANET LABS 2023(B)). Zudem gibt es noch verschiedene Asset Typen, wodurch die Produkte bspw. als „Analytic“, „Basic Analytic“ „Visual“ oder „Surface Reflectance“ erhältlich sind. Die verschiedenen Typen besitzen unterschiedliche Prozessierungen und sind an verschiedene Interessengruppen adressiert. Zum Training des Modells wurden Orthotiles vom Typ „Analytic“ gewählt. Hierbei handelt es sich um kalibrierte, orthorektifizierte (UTM) Multispektraldaten mit den obigen fünf Kanälen. Zusätzlich wurden die Daten für Sensorartefakte und Verschiebungen im Gelände korrigiert (PLANET LABS (2023(A)). Mittels mitgelieferter Metadaten lassen sich diese überlieferten Pixelwerte in Top of Atmosphere (TOA) Reflektanzen umrechnen. TOA Reflektanzen beziehen sich auf die Reflektanz von Sonnenstrahlen an der oberen Grenze der Atmosphäre. Im Gegensatz zu Reflektanzen an der Erdoberfläche (Surface Reflectance) sind TOA-Werte frei von der Wechselwirkung mit der Atmosphäre. Tritt Licht in die Atmosphäre ein, kann es auf dem Weg durch die einzelnen Luftschichten gestreut oder absorbiert werden. Hierdurch können sich optische Eigenschaften wie Farbwahrnehmung oder Lichtintensität verändern (ZHAI ET AL. 2022). Um diese Wechselwirkungen zu umgehen, wurde sich für TOA-Reflektanzen entschieden. Die Daten werden mit 16-bit und 5m GSD (Ground Sample Distance) an der Erdoberfläche ausgeliefert.

3.2 PlanetScope

Das PlanetScope Programm ist neben RapidEye eine weitere Erdbeobachtungsmission von Planet Labs. Seit 2014 wurden in mehreren Starts ungefähr 130 Dove Satelliten in die Erdumlaufbahn gebracht, die pro Tag einmal die gesamte Landoberfläche aufnehmen können. Die verbauten Instrumente haben sich von Dove Classic mit vier aufgenommenen Bändern (R-G-B-NIR) zu den neueren SuperDove Satelliten mit acht Bändern entwickelt. Seit März 2020 nehmen diese Satelliten neben RGBNIR auch „New Red Edge“, „Green 1“, „Coastal Blue“ und „Yellow“ auf und tragen damit zu einer Abdeckung eines größeren Wellenlängenbereichs bei. Je nach Sensor unterscheiden sich die aufgenommenen Szenegrößen von ca. 280km² (Dove Classic) bis hin zu 630km² (SuperDove) (PLANET LABS (2023(C)). Die gewünschten Produkte können wie bei RapidEye in unterschiedlichen Asset Typen geordert werden. Für die Vergleichbarkeit zu RapidEye wurde sich hier ebenfalls für Analytic entschieden. Anders als bei RapidEye, kann bei PlanetScope

nur zwischen sog. BasicScenes und OrthoScenes unterschieden werden. Die Prozessierungsstufen unterscheiden sich hierbei von L1B BasicScene und L3B für die OrthoScenes. Um nicht eigenständig zu orthorektifizieren oder radiometrisch nachbessern zu müssen, wurde sich hier für die OrthoScenes in L3B entschieden. Eine detaillierte Übersicht zu den unterschiedlichen Produkten mit deren Prozessierungsstufe ist Anhang 1 zu entnehmen (PLANET LABS, 2023(B)).

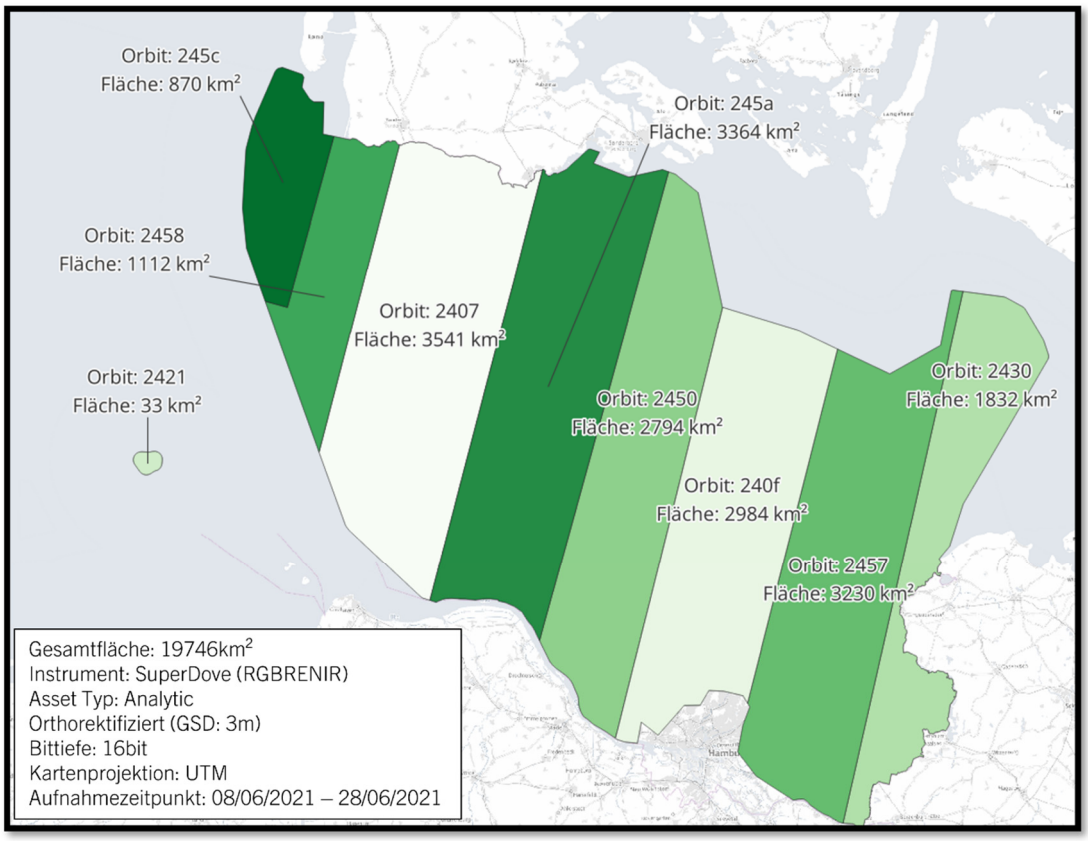


Abbildung 11: Übersicht über verwendete PlanetScope Daten. Dargestellt sind die verwendeten Orbits mit deren räumlicher Ausdehnung. Die Orbit IDs wurden von Planet Labs übernommen. Weiterhin wurden die Größe der Orbits in km² sowie generelle Eigenschaften (Asset Typ, Aufnahmezeitpunkt, Beobachtete Gesamtfläche, Bittiefe und Projektion) der PlanetScope Daten hinzugefügt (Eigene Darstellung).

Bei der Beschaffung der Daten mussten weiterhin einige Besonderheiten beachtet werden. Planet Labs bietet wissenschaftlichen Einrichtungen verschiedene Datenkontingente zum Bezug derer Produkte an. Neben der „Basic“ Variante mit 5000 km² pro Monat gibt es weitere Kontingentmöglichkeiten. In Anbetracht der ca. 23.000km² für die Abdeckung Schleswig-Holsteins wurde hier das monatliche Kontingent erhöht. Abbildung 11 gibt eine Zusammenfassung der Daten des Untersuchungsgebiets Schleswig-Holstein (SH). Für diese Arbeit wurden SuperDove Daten bezogen, die auf die Kanäle RGBRENIR beschränkt wurden. Für en Aufnahmezeitpunkt war es wichtig, möglichst wolkenfreie Bilder aus derselben Jahreszeit zu erhalten. Für das Jahr 2021 eignete sich der Monat Juni,

da in diesem Zeitrahmen die Wolkenbedeckung minimal war und somit der ohnehin kleine Datenbestand nicht noch weiter ausmaskiert werden musste.

Für die Abdeckung der gesamten Landesfläche SH waren die Daten von neun Orbits notwendig (Abb. 11). Auffällig hier ist die variierende Größe der einzelnen Polygone, die durch den Bezug der Daten als Komposit (um Datenkontingent zu sparen) entstanden. Der PSB.SD (SuperDove) Sensor nimmt innerhalb eines Orbits Szenen mit einer Größe von ca. 32,5km x 19,6km auf. Diese überlappen sich sowohl innerhalb desselben Orbits als auch zu den benachbarten Streifen. Bei der Erstellung eines Komposites/Mosaiks im Planet Explorer (dem Shop System von Planet Labs) werden die neueren Daten über die älteren Daten der Nachbarorbits gelegt. Das bedeutet das bei der Komposit Bildung immer die neusten Daten für ein spezielles Areal zur Verfügung stehen. Besitzen die benachbarten Orbits ältere Bilder, werden diese nicht berücksichtigt. Diese Eigenschaft den Komposit konnte visuell anhand der mitgelieferten Szenen als GeoJSON verifiziert werden. Zusätzlich zu den Szenen als GeoJSON und dem Komposit erhält die Datenlieferung noch XML-Dateien mit Metadaten zur jeweiligen Szene. Diese Metadaten enthalten neben Produktinformationen, wie Aufnahmezeitpunkt und geometrische Eigenschaften, auch Umrechnungskoeffizienten pro Band. Da es hier um Top of Atmosphere (TOA) korrigierte Daten handelt, lassen sich die überlieferten „Digital Numbers (DNs)“ in 16bit mittels Reflektanzkoeffizienten in TOA Reflektanzen umrechnen. Diese Koeffizienten variieren innerhalb eines Orbits nur minimal, weswegen sich zur Bildung des Mittelwerts pro Bildkanal entschieden wurde. Tabelle 3 zeigt die gemittelten Reflektanzkoeffizienten pro Orbit und Kanal.

Tabelle 3: Gemittelte Reflektanzkoeffizienten pro Bildkanal je Orbit. Die Koeffizienten werden zur Umrechnung von überlieferten Digital Numbers (DN) in TOA Reflektanzen benötigt (Eigene Darstellung).

Orbit	B (465-515nm)	G (547-583nm)	R (650-680nm)	RE (697-713nm)	NIR (845-885nm)
245c	$2{,}07 \times 10^{-5}$	$2{,}24 \times 10^{-5}$	$2{,}70 \times 10^{-5}$	$2{,}89 \times 10^{-5}$	$4{,}27 \times 10^{-5}$
2458	$2{,}05 \times 10^{-5}$	$2{,}23 \times 10^{-5}$	$2{,}68 \times 10^{-5}$	$2{,}86 \times 10^{-5}$	$4{,}24 \times 10^{-5}$
2430	$2{,}04 \times 10^{-5}$	$2{,}21 \times 10^{-5}$	$2{,}67 \times 10^{-5}$	$2{,}85 \times 10^{-5}$	$4{,}22 \times 10^{-5}$
2457	$2{,}05 \times 10^{-5}$	$2{,}22 \times 10^{-5}$	$2{,}67 \times 10^{-5}$	$2{,}86 \times 10^{-5}$	$4{,}23 \times 10^{-5}$
245a	$2{,}05 \times 10^{-5}$	$2{,}23 \times 10^{-5}$	$2{,}68 \times 10^{-5}$	$2{,}86 \times 10^{-5}$	$4{,}24 \times 10^{-5}$
2450	$2{,}05 \times 10^{-5}$	$2{,}22 \times 10^{-5}$	$2{,}67 \times 10^{-5}$	$2{,}86 \times 10^{-5}$	$4{,}23 \times 10^{-5}$
241a	$2{,}06 \times 10^{-5}$	$2{,}23 \times 10^{-5}$	$2{,}69 \times 10^{-5}$	$2{,}87 \times 10^{-5}$	$4{,}26 \times 10^{-5}$
2421	$2{,}05 \times 10^{-5}$	$2{,}22 \times 10^{-5}$	$2{,}67 \times 10^{-5}$	$2{,}86 \times 10^{-5}$	$4{,}23 \times 10^{-5}$
240f	$1{,}95 \times 10^{-5}$	$2{,}12 \times 10^{-5}$	$2{,}55 \times 10^{-5}$	$2{,}73 \times 10^{-5}$	$4{,}04 \times 10^{-5}$
2407	$1{,}96 \times 10^{-5}$	$2{,}12 \times 10^{-5}$	$2{,}55 \times 10^{-5}$	$2{,}73 \times 10^{-5}$	$4{,}04 \times 10^{-5}$

Diese Koeffizienten werden mit den DNs im jeweiligen Band multipliziert und ergeben die TOA Reflektanzen. Da das Modell initial auf TOA Reflektanzen von RapidEye trainiert wurde, wird durch diesen Schritt eine Vergleichbarkeit hergestellt.

3.3 Landbedeckungsmodell (LBM-DE)

Das digitale Landbedeckungsmodell (LBM-DE) wird seit 2009 als Standardprodukt des Bundesamts für Kartographie und Geodäsie (BKG) geführt. Es ist in Landnutzung und Landbedeckung unterteilt und beschreibt zugehörige Geometrie flächendeckend für das gesamte Bundesgebiet im Vektorformat. Zielsetzung hierbei ist, den Zustand der Umwelt zum Erfassungszeitpunkt festzuhalten. Die Erfassung und die damit verbundene Aktualisierung erfolgt 3-jährig mit Bezug auf ein bestimmtes Referenzjahr. Durch Vergleich der verschiedenen Referenzjahre wird der Benutzer befähigt, kurz- und längerfristige Änderungen der Landschaft wahrzunehmen und diese durch räumliche Analysen zu untersuchen. Die Aktualisierung des LBM-DE unterliegt hohen Qualitätsstandards mit DIN-basierten Qualitätskontrollen im Stichprobenverfahren durch Mitarbeiter des BKGs. Dies gewährleistet saubere Topologien und verhindert Überlappungen und Lücken. Um die hohen Qualitätsstandards zu erreichen, werden eine Vielzahl von Eingangsdaten verwendet, auf deren Basis die Aktualisierung vorgenommen wird (BKG, 2023). Die Grundlage bilden hier Satellitenbilddaten als Rasterdatenquelle und das ATKIS Basis-DLM als Vektordatenquelle sowie das LBM-DE des letzten Aktualisierungszyklus, um Veränderungen der Landschaft zu erkennen. Für das Bezugsjahr 2018 wurden beispielsweise das ATKIS Basis-DLM mit Stand 2017 und Satellitenbilddaten (RapidEye und Sentinel-2) aus den Jahren 2017 und 2018 genutzt. Weiterhin wurden digitale Orthophotos der Landesvermessung herangezogen. Auf dieser Grundlage erfolgte die Erstellung des LBM-DE 2018. Die breite Datengrundlage sowie die hohen Standards bei der Aktualisierung durch Mitarbeiter des BKGs machen das LBM-DE zu einem präzisen Modell mit einer Vielfalt an Anwendungsbereichen. Diese erstrecken sich vom Umweltmonitoring und der damit verbundenen Beobachtung von Zustandsveränderungen in Ökosystemen bis hin zur Klimafolgenforschung und dem Einfluss von Landbedeckungsänderungen und Anpassungsstrategien als Zeichen des Klimawandels. Ein prominentes Beispiel ist die Ableitung des CORINE Land Cover (CLC) Datensatzes für das Bundesgebiet Deutschlands. Hierbei muss generalisiert werden, da sich die Mindestkartierfläche (1ha) und Mindestkartierbreite

(15m) des LBMs von den entsprechenden Kartiereinheiten des CLC Datensatzes unterscheiden (BKG, 2023). Speziell die hohe Klassenvielfalt trägt zur vielfachen Nutzbarkeit des LBMs bei. Tabelle 4 zeigt die 31 Landbedeckungsklassen des LBMs, eingeteilt in sieben Hauptkategorien.

Tabelle 4: Übersicht über die unterschiedlichen Klassen des Landbedeckungsmodells Deutschland (LBM-DE). Die Klassen sind in die sieben Hauptkategorien eingeteilt: Bebaute Flächen (A), Landwirtschaft (B), Grünland (C), höhere Vegetation (D), sowie vegetationslose (E), feuchte (F) oder Wasserflächen (G). Zusätzlich wurde die Beschreibung (2.Spalte), der BKG interne Code pro Landbedeckungsklasse (3. Spalte) sowie der im Modell vergebene Code (4. Spalte) angegeben. Die Codes sind hier als ID der jeweiligen LB zu verstehen (modifiziert nach BKG, 2023).

Kat	Landbedeckungsklasse	Code(BKG)	Code (M)
A	Bebauung	B110	1
	Anlagen	B121	2
	Versiegelte gebäudelose Flächen	B122	3
	Mischflächen	B242	4
B	Ackerland	B211	5
	Weinbau	B221	6
	Obst- und Beerenobst	B222	7
	Hopfen	B224	8
C	Homogenes Grünland	B231	9
	Inhomogenes Grünland	B321	10
	Grasland mit Bäumen (<50%)	B233	11
D	Zwergsträucher (Heide)	B322	12
	Büsche, Sträucher	B324	13
	Aufforstung	B310	14
	Laubbäume	B311	15
	Nadelbäume	B312	16
	Nadel- und Laubbäume	B313	17
E	Sand, Steine, Erde	B330	18
	Fels	B332	19
	Brandfläche	B334	20
	Schnee und Eis	B335	21
F	Sumpf	B411	22
	Moor	B412	23
	Sumpf mit Büschen/Bäumen	B413	24
	Moor mit Büschen/Bäumen	B414	25
G	Watt	B423	26
	Wasserlauf	B511	27
	Wasserfläche	B512	28
	Lagune	B521	29
	Mündungstrichter	B522	30
	Offenes Meer	B523	31

Diese erstrecken sich von bebauten Flächen (A) und Landwirtschaft (B) über Grünland (C) und höherer Vegetation (D) bis hin zu vegetationslosen (E), feuchten (F) oder Wasserflächen (G). Weiterhin gibt es für die darin enthaltenen Landbedeckungen (LB) verschiedene Landnutzungen (LN). Beispielsweise besitzt die Klasse „Versiegelte gebäudelose Flächen“ LN-Klassen wie „versiegelte Flächen bei Häfen“ (N123), „versiegelte Flächen bei Abbauflächen“ (N131) oder „versiegelte Flächen bei Deponien“(N132). Da es sich in dieser Arbeit um eine pixelbasierte Segmentierung handelt, ist diese Unterscheidung nicht ohne weitere Schritte möglich. Denkbar wären bspw. anschließender räumliche Analyse der klassifizierten versiegelten Flächen und ein Verschnitt mit z.B. Deponien die im digitalen Landschaftsmodell (Basis-DLM) geführt werden. Solche Schritte werden innerhalb dieser Arbeit nicht unternommen, und die Unterscheidung wird auf die spektralen Signale der Pixel innerhalb der verschiedenen LB-Klassen beschränkt. Zur Tabelle 4 sei noch angemerkt, dass neben den Klassenkürzeln des BKGs (BXXX) ebenfalls ein ID seitens des Modells vergeben wurde (1-31). Die Zuordnung kann Tab. 4 entnommen werden.

Zur Interpretation der Klassifizierungen dieser Arbeit und deren weiterer Verwendbarkeit ist ein vertiefter Blick in den aktuellen Aktualisierungsprozess des LBM-DE notwendig. Das Hauptverwendungsziel des LBM-DE ist der deutsche Beitrag zum europaweit harmonisierten Landbedeckungs- und Landnutzungsdatensatz, CORINE Land Cover (CLC). Seit der ersten Ableitung 2009 angelehnt an die CLC-Nomenklatur, wurde 2012 eine Trennung von Landbedeckung und Landnutzung eingeführt. Das ATKIS Basis-DLM und dessen geführte Informationen für Landbedeckung und Landnutzung stellten hier die Basis für die Überführung in das neue Klassensystem des LBM 2012 (BKG 2023). Seit dieser initialen Ableitung werden die bestehenden Polygone des LBMs auf Veränderung untersucht und angepasst. Es werden sog. Vegetations- und Versiegelungsgrade (VEG_AKT und SIE_AKT) für jedes Objekt aus vorhandenen Bilddaten bestimmt, welche als attributive Zuordnung im Nachgang zur LBM-Klassenzuordnung notwendig sind. Wichtig ist hierbei die bereits erwähnte Mindestkartierfläche von 1ha, die sich nur auf neu zu erfassende Objekte bezieht. Somit sind bestehende Objekte aus der Erstableitung des ATKIS Basis-DLM, sofern keine Änderungen notwendig sind, davon unberührt. Eine Besonderheit bilden Schnittrestflächen (min. 0,2ha), die durch eventuelle Ausbreitung umliegender LB entstehen können. Sollte die ursprüngliche Fläche unter 0,2ha fallen, erfolgt eine Bewertung der gesamten Fläche mittels Mehrheitsprinzip (BKG 2023). Eine

Ausweisung der zu generierenden Trainingsgebiete für den aktuellen Aktualisierungsprozess findet sich in Anhang 9. Hierbei spielen die vorhandenen Objekte der LB-Klassen (mit verschiedener LN) sowie die Neuberechnung von Versiegelung- und Vegetationsgrade sowie spektraler Indizes (z.B. NDVI) eine Rolle.

3.4 Aufbau des Modells

In dieser Arbeit wird ein sog. Encoder-Decoder System verwendet. Als Encoder dient ein ResNet50, welches Merkmale aus den Eingangsdaten extrahiert, um sie anschließend an den Decoder (hier FPN) zu übergeben. Abb. 12 zeigt die detaillierte Modellarchitektur. Auf der linken Seite befindet sich das ResNet50 mit verschiedenen Faltungsblöcken (C1 bis C5) und auf der rechten Seite das FPN mit den Faltungsblöcken (P2-P5). Am Ende jedes Faltungsblocks des ResNet50 werden die resultierenden Feature Maps via Skip Connections an das FPN übergeben. Hierfür sind die gleiche räumliche Ausdehnung sowie die gleiche Anzahl an Filtern essenziell. Die Brüche an den jeweiligen Blöcken im Encoder und Decoder zeigen, dass die räumliche Ausdehnung der Daten bereits angepasst ist (z.B. C2 und P2= 1/4 der Eingangsgröße, C3 und P3 = 1/8 der Eingangsgröße). Durch die Skip Connection wird durch ein 1x1 Faltung die Filteranzahl auf 256 Filter erhöht, bzw. verringert (je nach Filteranzahl der finalen Faltungsschicht innerhalb von C2 bis C5).

Als Eingangsdaten werden 5 Kanalbilder (RGBRENIR, angelehnt an die spektralen Kanäle von RapidEye und PlanetScope) mit unterschiedlichen Bilddimensionen genutzt.

Diese Dimensionen können Bildgrößen von 96x96, 192x192, 224x224, 320x320, 352x352, 384x384, 416x416, 512x512 Pixel einnehmen. In dieser Arbeit wurde eine Kachelgröße von 320x320 verwendet. Fließen die Eingangsdaten im Minibatch durch den Encoder werden Sie zunächst gefaltet (7x7 Kernel) und die Bildgröße durch die Schrittweite von zwei halbiert. Als Ausgabe ergeben sich 160px x 160px Bilder mit 64 Filtern (visualisiert unter der jeweiligen Schicht in Abb. 12 C1). Wie in allen Faltungsschichten (orange Blöcke) entscheidet das Modell selbst, welche Merkmale es lernt, und welche Filter es dafür benötigt. Da es sich um Transfer Learning handelt, nutzt das Modell initial die Gewichte des vortrainierten Modells auf Basis der RapidEye Daten. Nachdem die Daten normalisiert (nach jeder Faltung) wurden, werden der Minibatches durch einen MaxPooling Layer (3x3 Kernel mit Schrittweite 2 (lila Block)) gesendet, der zur weiteren Reduktion der Bilddimensionen beiträgt.

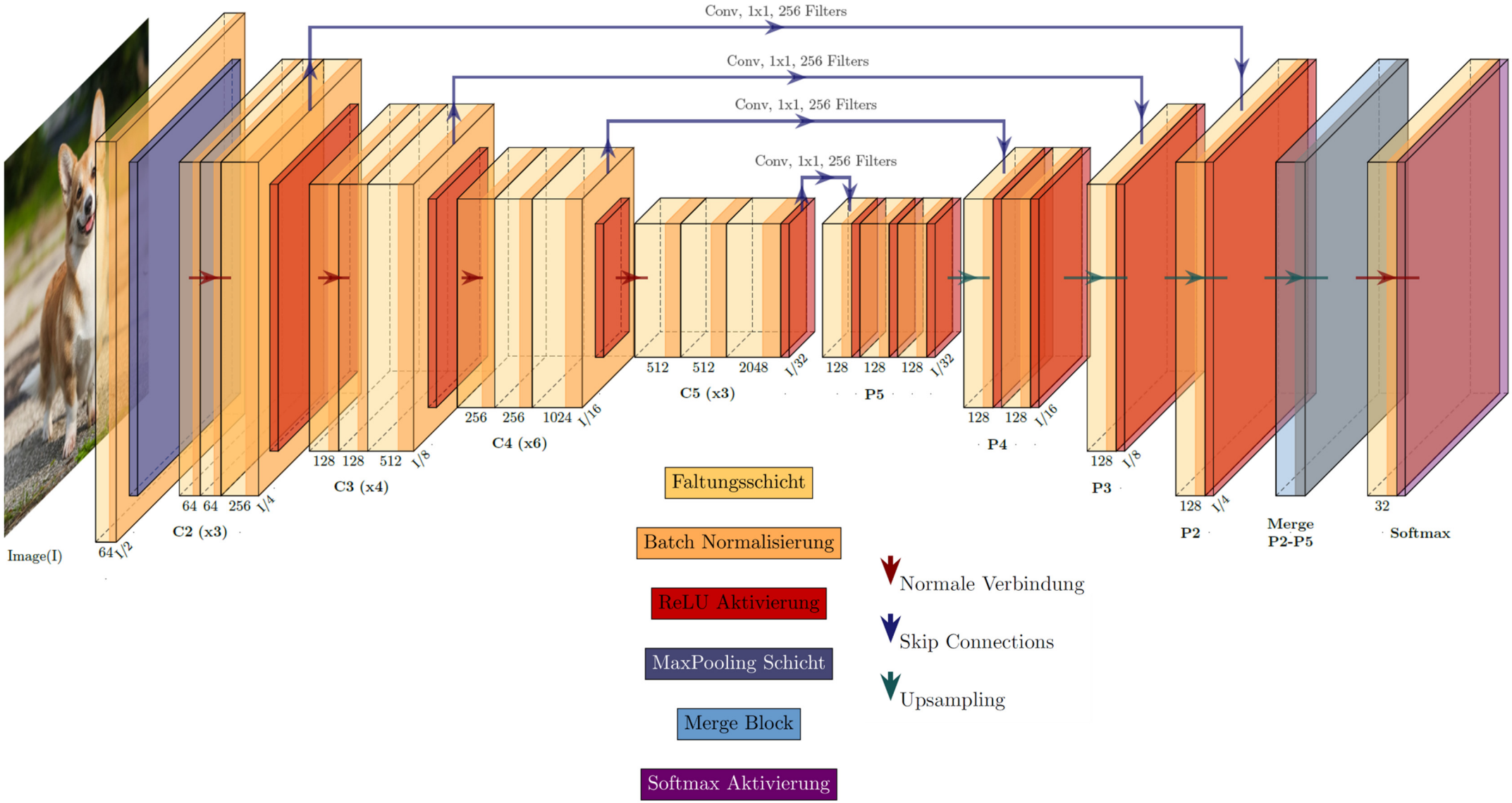


Abbildung 12: Schematischer Aufbau des verwendeten neuronalen Netzes. Abgebildet ist ein Encoder-Decoder System mit ResNet50 als Convolutional Backbone und einem FPN als Decoder. Die Faltungsschichten sind in gelb dargestellt und die einzelnen Faltungsblöcke haben Beschriftungen (C2 bis C5). Oberhalb der Grafik sind in blau die Skip Connections eingezeichnet, die den Übertrag der Feature Maps der Faltungsblöcke (C2 bis C5) zu deren Pendant innerhalb des FPNs (P2 bis P5) skizzieren. Weiterhin ist die Dimensionsreduktion des Eingangsbildes I an den jeweiligen Schichten vermerkt (z.B. C2 = I/4) (Eigene Darstellung).

Die nachfolgenden Faltungsblöcke C2 bis C5 erhalten jeweils drei Faltungen und eine Verdoppelung der Anzahl der Filter pro Block (C2 = 64|64|256; C3 = 128|128|512; C4 = 256|256|1024 und C5 = 512|512|2048). Zusätzlich wird die Bilddimension stetig halbiert. Sowohl die Reduktion der Bilddimension als auch die wachsende Anzahl an Filter wird durch kleinere und breitere Blöcke visualisiert (siehe Abb. 12). Ebenfalls ist der Abbildung zu entnehmen, wie häufig ein Faltungsblock ausgeführt wird (C2 = 3x; C3 = 4x, C4 = 6x, C5 = 3x). Wie in Kapitel 2.3.4 beschrieben, sind Skip Connections für eine stabile Backpropagation innerhalb der ResNets essenziell, benötigen aber gleiche Bilddimensionen. Bei C2 ist dies durch vorherige Dimensionsreduktion im MaxPooling Layer noch gewährleistet. Hier muss lediglich die Anzahl der Filter an die Filter der letzten Faltungsschicht im Block angepasst werden. Es sind also nur Identity Blocks (vgl. Kapitel 2.3.4) mit Anpassung der Filter ohne Veränderung der Bilddimensionen notwendig. Ab Schicht C3 erfolgt innerhalb des ersten Durchlaufs der Faltungsblocks eine Reduktion der Bilddimensionen durch Veränderung der Schrittweite (2) der zweiten Faltung. Aus diesem Grund kann das Ergebnis der vorherigen Schicht nicht mit dem Ergebnis der ersten Faltung verrechnet werden und es wird ein Convolutional Block benötigt. Im Convolutional Block werden nun neben der Filteranzahl auch die Bilddimensionen angepasst. C3 besteht demnach aus einem Convolutional Block und drei Identity Blocks (C4 = 1/5; C5 = 1/2). Nach jeder Faltungsschicht werden die Daten zur Vergleichbarkeit normalisiert und am Ende eines Durchlaufs des Faltungsblocks laufen die Ergebnisse durch die Aktivierungsfunktion ReLU (rote Blöcke).

Am Ende der Faltungsblöcke C2 bis C5 werden die entstandenen Feature Maps an die jeweilige Schicht (P2-P5) im FPN übergeben. Nach Abschluss der Feature Extraktion im Encoder, der hier als Top-Down Pathway agiert (vgl. Kap. 2.3.5), erfolgt der Upsampling Vorgang im Decoder (Bottom-Up Pathway). Ausgehend von den Feature Maps mit den kleinsten Bilddimensionen, aber den bedeutungsvollsten Merkmalen (höchster semantischer Wert), wird nun eine Verbindung dieser mit den höher aufgelösten Feature Maps hergestellt. Beginnend mit P5 durchlaufen die Feature Maps aus C5 drei Faltung mit Aktivierungen. Hierbei werden die Filter von 256 auf 128 reduziert. Diese Anpassung ist notwendig für das Zusammenführen der entstandenen Feature Maps am Ende. Sind diese Faltungen abgeschlossen, beginnt das Upsampling um den Faktor zwei. Unter Hinzunahme der nächsten Nachbarn wird das Ergebnis von P5 mit den eingehenden Feature Maps aus P4 verbunden. Die höhere und genauere Auflösung unterstützt das Upsampling

und führt zur genaueren Lage extrahierter Features. Dieser Prozess wird rekursiv für die Schichten P4 bis P2 wiederholt und die Feature Maps aus den jeweiligen lateralen Verbindungen aufaddiert. In jeder Schicht erfolgen ebenfalls Faltungen mit Aktivierungen, um Effekte des Upsamplings zu minimieren. Nach P2 werden die Feature Maps auf die Bilddimensionen der Eingangsdaten gebracht und final gefaltet. Bei dieser Faltung handelt es sich um Reduktion der Filter pro Feature Map von 128 auf 32. Die abschließende Aktivierungsfunktion Softmax (lila transparente Box) weist die Objekte nun der wahrscheinlichsten Klasse zu. Die 32 bezieht sich hier auf die 31 Landbedeckungsklassen des LBMs plus die NoData Werte.

Die Architekturen des Encoders und Decoders sind hier den Vorgaben aus PyTorch übernommen werden. Speziell beim FPN zeigt die Literatur ggf. leichte Abweichungen in der Anzahl der Faltungen pro Schicht. Die grundsätzliche Architektur ist allerdings identisch. Der verwendete Optimierer ist Adam und als Loss Funktion zur Minimierung des Verlusts wurde CrossEntropy verwendet (vgl. Kap. 2.3.1). Bei der Verlustfunktion wird zusätzlich sog. Label Smoothing angewandt. Beim Label Smoothing handelt es sich um eine Regularisierungsmethode um die Robustheit von Modellen zu verbessern und die Anfälligkeit gegenüber Overfitting zu reduzieren. Hierbei werden starre Zuordnungen der Ground Truth Labels von 0 und 1 durch sanftere Wahrscheinlichkeitsverteilungen ersetzt (MÜLLER ET AL. 2019). Zur Verdeutlichung kann man sich eine drei Klassen Klassifizierung im One Hot Encoding vorstellen. Demnach wären die Ground Truth Labels der verschiedenen Instanzen entweder [1,0,0], [0,1,0] oder [0,0,1]. Würde man nun die Labels mit einem Smoothing von 0.1 (10%) glätten, hätte die korrekte Klasse eine Wahrscheinlichkeit von 90% und die restlichen 10% werden auf die übrigen Klassen aufgeteilt. Für die erwähnten Labels bedeutet dies [0.9,0.05,0.05], [0.05,0.9,0.05] oder [0.05,0.05,0.9]. Die Verwendung des Labels Smoothing hat zur Steigerung der Treffergenauigkeit von DL-Modellen in unterschiedlichsten Anwendungsbereichen geführt, unter anderem Bildklassifizierung (MÜLLER ET AL. 2019).

3.5 Vorbereitung der Daten

Die Analyse in dieser Arbeit erfolgt als Kreuzvalidierung. Der geringe Datenbestand kann den Domänenübertrag von RapidEye zu PlanetScope trotz der Nutzung eines vortrainierten Modells potenziell erschweren. Aus diesem Grund soll der Datensatz nicht weiter dezimiert werden. Bei der Kreuzvalidierung wird iterativ über die Orbits gegangen, sodass jeder Orbit einmal Testgebiet war. Mittelt man die Treffergenauigkeit pro Orbit in Bezug auf deren absolute Fläche erhält man eine Gesamtgenauigkeit für Schleswig-Holstein. Abseits des Testorbits werden die restlichen Orbits randomisiert als Trainings- und Validierungsdaten im Verhältnis 70/30 zugeteilt. Wie in Kapitel 2 beschrieben, lernt das Modell von Trainingsdaten und modifiziert verwendete Hyperparameter auf Basis der Validierungsdaten. Aus diesem Grund ist Aufteilung wichtig und führt zu robusteren Modellen.

Zu Beginn des Trainingsprozesses müssen die Daten in die für die Modellarchitektur passende Form gebracht werden. Für die Semantische Segmentierung sind Labels notwendig. Aus diesem Grund wird das LBM aus dem Jahr 2021 vom Vektorformat ins Rasterformat transformiert. Hierdurch wird eine flächendeckende Rastermaske erstellt, wodurch jedem Pixel der Trainingsdaten ein eindeutiges Label zugewiesen werden kann. Dies ist für die pixelbasierte Segmentierung essenziell. Vor dem eigentlichen Training werden alle Orbits (außer des jeweiligen Testorbits) in 320px x 320px Rastertiles gekachelt. Gleichzeitig erfolgt die Aufteilung in 70% Trainingsdaten und 30% Validierungsdaten. Tabelle 5 zeigt die Kachelanzahl für Trainings- und Validierungsdaten für die Orbits von Schleswig-Holstein.

Tabelle 5: Übersicht über Trainings- und Validierungskacheln nach Orbit (Eigene Darstellung).

Orbit	Trainingskacheln	Validierungskacheln	Gesamtanzahl
245c	15060	6455	21515
2457	13280	5692	18972
245a	12659	5426	18085
2450	13620	5838	19458
2458	14884	6380	21264
240f	13079	5606	18685
2430	14334	6144	20478
2421	15680	6721	22401
2407	13041	5589	18630

3.6 Trainingsprozess und Hyperparameteroptimierung

Nachdem die Daten vorverarbeitet, gekachelt und in Trainings-, Test- und Validierungsdaten geteilt wurden, beginnt der eigentliche Trainingsprozess. Während des Trainings gibt es mehrere Hyperparameter, die die Modellgüte und -performance beeinträchtigen können. Hyperparameter werden zu Beginn des Trainings festgelegt und bleiben konstant. Eine Ausnahme bildet hier die Lernrate, die über weitere Hyperparameter im Laufe des Trainings angepasst werden kann. Unter Verwendung von Adam zur Optimierung erfolgt in dieser Arbeit nach 10 Epochen eine Halbierung der Lernrate. Eine schrittweise Anpassung der Lernrate ist gängige Praxis im Deep Learning Kontext. Schrumpfende Lernraten tragen in späteren Schichten zu stabileren Modellen und besserer Konvergenz in Richtung des globalen Minimums bei. Da die Lernrate bestimmt, wie schnell das Modell im Gradientenabstieg in Richtung Minimum konvergiert, hat diese starken Einfluss auf die Rechenzeit. Tabelle 6 gibt einen Überblick über die Anpassungen im Rahmen der Hyperparameteroptimierung. Die Optimierung erfolgte auf dem identischen Testorbit (2457) und es wurde pro Trainingslauf je ein Hyperparameter geändert, um dessen direkte Auswirkung dokumentieren zu können. Das Training gibt die besten zwei Modelle mit dem niedrigsten Verlust für die Validierungsdaten (Val_Loss) aus. In Abbildung 13 ist Val_Loss für die Trainingsläufe der Hyperparameteroptimierung visualisiert.

Tabelle 6: Hyperparameteranpassung auf Testorbit 2457. Verwendete Hyperparameter sind die Anzahl der Epochen, die Lernrate und die Batch Size.

Version	Epochen	Lernrate	Testszene	Genauigkeit	Batch Size
1	20	0,001	2457	82,81	16
2	40	0,001	2457	83,64	16
3	60	0,001	2457	84,26	16
4	60	0,0001	2457	83,7	16
5	60	0,01	2457	76,13	16
6	60	0,001	2457	83,8	16
7	80	0,001	2457	83,95	16
8	60	0,001	2457	84,09	32
9	60	0,001	2457	83,8	8

Die Spalte Version kann hier als ID der Trainingsläufe angesehen werden und dient zum besseren Vergleich. Die geeignete Wahl der Hyperparameter wird durch Interpretation des Trainingsverlust und des Validierungsverlusts getroffen. Während des Trainings wird

der Verlust auf Basis der Trainingsdaten weiter optimiert und minimiert. Optimalerweise sinkt der Validierungsverlust ebenfalls, was ein Indiz für eine gute Performance auf ungesehenen Daten ist. Sollte der Validierungsverlust im Laufe des Trainings wieder ansteigen, der Trainingsverlust hingegen weiter fallen, besteht die Gefahr des Overfittings.

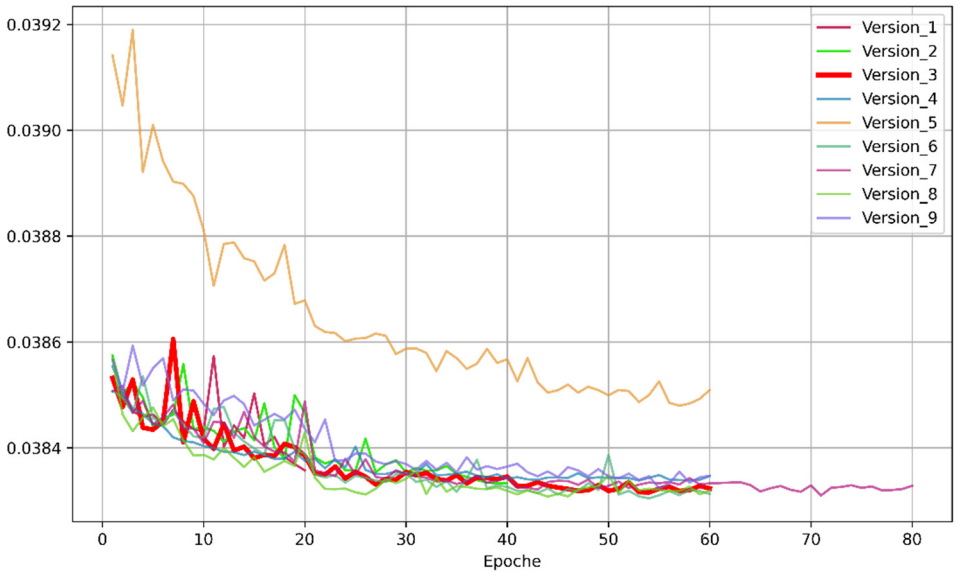


Abbildung 13: Validierungsloss für die verschiedenen Testläufe innerhalb der Hyperparameteroptimierung (Eigene Darstellung).

Aus diesem Grund werden in dieser Arbeit die besten Modelle auf Basis des Val_Loss ausgegeben. Der abgeflachte Val_Loss in Kombination mit abflachendem Trainingsverlusts (siehe Anhang 2) ist ein Indiz für die Erreichung eines Minimums. Dieses wird maßgeblich von den verwendeten Lernraten beeinflusst. Ausgenommen Version 5 mit höherer Lernrate, zeigen die anderen Versionen ähnliche Werte für den Val_Loss (siehe Abb. 13) und Train_Loss (Anhang 2). Selbst die niedrigere Lernrate (0,0001) von Version 4 zeigt keine weitere Senkung der Verluste. Die höhere Lernrate von 0,01 (Version 5) verdeutlicht eindrücklich den stärkeren initialen Abfall der Verlustfunktion, zeigt aber auch, dass sich die Funktion auf einem höheren Niveau einpendelt. Dies führt zur Übernahme einer Lernrate von 0,001.

Neben der Lernrate wurden ebenfalls die Epochenanzahl und die Batch Size angepasst. Die geeignete Anzahl der Epochen pro Training definiert sich durch die Zeit, die ein Modell benötigt, um die Verlustfunktion zu minimieren und dem Zeitpunkt, ab dem keine Verbesserung mehr erreicht wird. Letzteres wird durch den Vergleich von Version 3 mit

60 Epochen und Version 7 mit 80 Epochen hervorgehoben. Aufgrund anhaltender Plateauwirkung lohnt sich das Training von 20 weiteren Epochen hier nicht. Ebenfalls wurden 20 (Version 1) und 40 (Version 2) Epochen getestet (siehe Tabelle 6). Trotz marginalem Abfall des Validierungsverlusts nach Epoche 10, zeigen die Testläufe kaum Genauigkeitssteigerung (vgl. Tab. 6, Version 1 bis 3). Abb. 13 zeigt bei Version 3 (Abb. 13) nach Epoche 10 nur eine minimale Validierungsverlustreduktion. In diesem Hinblick wurde eine Kreuzvalidierung mit 10 Epochen für das gesamte Bundesland Schleswig-Holstein durchgeführt. Bei einem Sechstel der Zeit erreicht man hier eine Gesamtgenauigkeit von 81,49%. Dies ist nur rund ein Prozent schlechter als der 60 Epochenlauf aus Kapitel 4.3 (82,27%). Hier ist im Deep Learning immer abzuwiegen, ob sich die zusätzliche Trainingszeit für marginale Verbesserung lohnt. In dieser Arbeit wurde sich für die Übernahme von 10 Epochen entschieden.

Als letzten Hyperparameter dieser Optimierung wurde die Batch Size variiert und deren Einfluss beobachtet. Die Batch Size gibt an, wie viele Bildkacheln dem Modell in einem Zuge präsentiert werden. Bei 18972 Bildkacheln für Testszene 2457 (vgl. Tab. 5) ergeben sich bei einer Batch Size von 16, 1186 Schritte (830 Training und 356 Validierung) pro Epoche. Die Veränderung der Batch Size hat zu keiner nennenswerten Veränderung des Validierungsverlusts geführt (vgl. Version 6, 8 und 9, Tab. 5). Bezogen auf die Gesamtgenauigkeit im anschließenden Test wurde eine Batch Size von 16 angenommen. Batch Sizes mit Werten von über 32 waren in der genutzten Hardwareumgebung nicht zu prozessieren.

Nach Wahl der Hyperparameter erfolgt das eigentliche Training. Für das Training werden als erstes Bildkanalstatistiken (Mean, Median, Min, Max, Std) und Gewichte für die LB-Klassen berechnet. Die Tests haben gezeigt, dass sowohl die Statistiken als auch die Gewichte über dem gesamten Datensatz berechnet werden sollten. Da Schleswig-Holstein eine regional sehr heterogene Landbedeckung aufweist, treten manche Klassen nur in speziellen Regionen der Landesfläche auf. Sind diese Bereiche Teil der Testorbits (insofern diese bei der Berechnung der Statistiken und Gewichte exkludiert waren) wurde diese schlechter erkannt, was maßgeblich Einfluss auf die Güte des Modells hatte. Eine entscheidende Klasse, die regional zu starken Unterschieden in den Bildkanalstatistiken führt, ist B423 (Watt), da diese lediglich im Westen des Landes auftritt und daher nicht in allen Orbits vorhanden ist. Solche Klassenverteilungen müssen ebenfalls bei der Er-

stellung der Gewichte und damit dem Einfluss einzelner Klassen auf die Gesamtgüte berücksichtigt werden. In Schleswig-Holstein liegt ein Klassenungleichgewicht vor. Klassen wie B211 (Ackerland: 34,85%), B231 (Homogenes Grünland: 21,36%) und B523 (Offenes Meer: 12,65%) machen fast 70% der betrachteten Fläche aus. Im Verhältnis dazu gibt es sehr viele Klassen mit flächenhaftem Vorkommen von unter 1% (siehe Abb. 14). Für diese unterrepräsentierten Klassen ist ein Ausbalancieren über alle Klassen notwendig, sodass alle Klassen nahezu gleichmäßig berücksichtigt werden und keine Verzerrung in Richtung häufiger auftretenden Klassen auftritt.

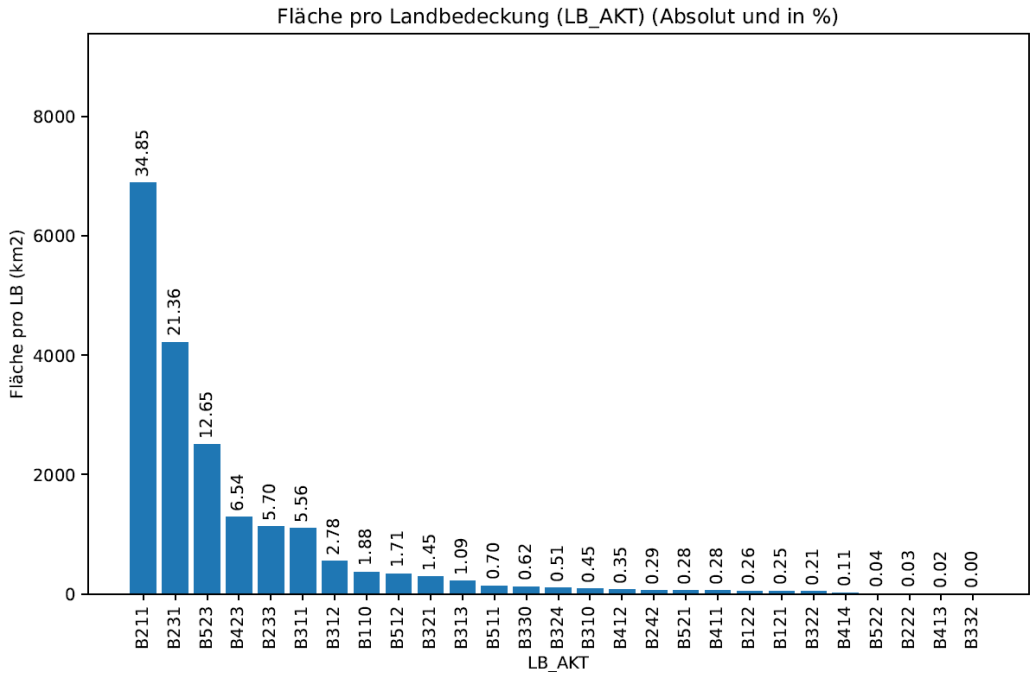


Abbildung 14: Fläche pro Landbedeckung im Untersuchungsgebiet Schleswig-Holstein. Aufgetragen sind die Landbedeckungsklassen des LBM-DE. Zusätzlich zur absoluten Fläche wurde deren prozentualer Anteil an der Gesamtfläche hinzugefügt (Eigene Darstellung).

In dieser Arbeit handelt es sich um eine pixelbasierte Segmentierung. Für die Bildung der Gewichte pro LB-Klasse werden die zugehörigen Pixel jeder LB gezählt. Weiterhin wird das 30% und das 70% Perzentil ausgegeben. Zur Balancierung werden Klassen, die unterhalb des 30-% Perzentils liegen auf Selbiges angehoben und präsentere Klassen oberhalb des 70-% Perzentils auf das 70-% Perzentil gesenkt. Anschließend erfolgt eine Normierung der angepassten Gewichte, bei der die inversen Werte der angepassten Gewichte durch die Summe der angepassten Gewichte geteilt wird. Nach Inversion erhalten Klassen mit niedrigeren Gewichten höhere inverse Gewichte und umgekehrt. Dies ist ein wichti-

ger Bestandteil der Balancierung der Klassen. Die normierten Gewichte werden abschließend noch durch Division der individualen Gewichte durch deren Summe skaliert und in einen Tensor zur weiteren Verarbeitung in PyTorch transformiert.

Nach der Ermittlung der Statistik und der Klassengewichte beginnt der eigentliche Trainingsprozess. Im Trainingsprozess greifen nun die festgesetzten Hyperparameter. Demnach erfolgt in dieser Arbeit ein Training über 10 Epochen bei einer Batch Size von 16 und einer Lernrate von 0,001. Während des Trainings sind im Modell verschiedene Augmentierungen implementiert. Hier werden zwischen Spiegelung (p = 0,5) und Verschiebung, Skalierung und Rotation (p= 0,3) unterschieden. Die Wahrscheinlichkeit (p) gibt hier an, wie hoch die Chance ist, dass besagte Transformation auf ein Bild angewendet wird. Hierbei sind mehrere verkettete Transformationen möglich. Diese Verfahren sollen dem Modell helfen, durch Variation der Trainingsdaten, besser auf ungesehenen Daten zu generalisieren. Die Transformationen der Eingangsdaten werden durch ein Dropout (p=0,2) unterstützt. Der Dropout schaltet mit festgesetzter Wahrscheinlichkeit einzelne Neuronen innerhalb des Trainingsprozesses aus. Hierdurch wird das Modell robuster und weniger abhängig von einzelnen Neuronen bzw. die Kombination einzelner Neuronen.

Neben dem „Lernen“ auf Basis der Trainingsdaten und der Minimierung des Verlusts ist die Performance auf den ungesehenen Daten des Validierungsdatensatzes entscheidend. Das Ergebnis des Trainings sind die besten zwei Modelle mit dem niedrigsten Validierungsverlust. Auf Basis dieser Modelle kann im Anschluss die Testszene segmentiert und diverse Gütemetriken ausgegeben werden. Diese Metriken helfen die Modellperformance in unterschiedlichen Aspekten zu bewerten.

Sowohl im Machine Learning als auch im Deep Learning gibt es eine Vielzahl an Metriken, die zur Gütebewertung der entwickelten Modelle genutzt werden. Dieses Feld unterliegt der ständigen Weiterentwicklung. Nichtsdestotrotz haben sich einige Metriken etabliert und werden sehr häufig zur Modellbewertung hergenommen (TERVEN ET AL. 2023). In Tabelle 7 sind die in dieser Arbeit verwendeten Metriken gezeigt.

Tabelle 7: Übersicht der verwendeten Metriken zur Bewertung der Gesamtgüte und klassenspezifischer Genauigkeiten. Aufgetragen sind die Beschreibung sowie die Berechnungen pro Metrik. Die Ausweisung dieser Metriken erfolgt bei pixelbasierten Ansätzen durch Angabe von Verhältnissen zwischen True Positives (TP), True Negatives (TN), False Positives (FP) und False Negatives (FN) in Abhängigkeit der jeweiligen Metrik.

Metrik	Berechnung	Beschreibung
(Treffer-) Genauigkeit	$\frac{TP + TN}{TP + TN + FP + FN}$	Gibt Verhältnis der richtig klassifizierten Pixel zu Gesamtanzahl an Pixel.
Präzision	$\frac{TP}{TP + FP}$	Gibt Auskunft über die tatsächlich positiven Fälle innerhalb der als positiv klassifizierten Fälle.
Recall /Sensitivität	$\frac{TP}{TP + FN}$	Richtig vorgesagte Positive zur Gesamtzahl der positiven Fälle pro Klasse
F1 - Score	$2\ x\ \frac{Präzision\ x\ Recall}{Präzision + Recall}$	Harmonisches Mittel aus Präzision und Recall

3.7 Verwendete Software

Während der Arbeit mit neuronalen Netzen gibt es zahlreiche Schritte, die neben dem eigentlichen Training des Netzes beachtet und effizient abgearbeitet werden müssen. Hierbei trägt die verwendete Software und deren Implementierung maßgeblich zur Geschwindigkeit der Abarbeitung bei. Im folgenden Kapitel wird die verwendete Programmiersprache mit zugehöriger Entwicklungsumgebung dargestellt. Weiterhin erfolgt eine Darstellung der Python Bibliotheken und weiterer Software, welche zur Vorverarbeitung, dem Training und der Darstellung der Ergebnisse verwendet wurden.

Python: Sowohl für die Arbeit mit Geodaten als auch das Training von neuronalen Netzen hat sich Python als eine der am häufigsten eingesetzten Programmiersprachen etabliert. In der Arbeit wird Python in Version 3.10. (2021) verwendet. Bei Python handelt es sich um eine interpretierte und u.a. objektorientierte Programmiersprache (ebenfalls prozedural oder funktional möglich). Sie ist einfach zu lesen und stammt von der Python Software Foundation. Durch die breite Community und die stetige Weiterentwicklung bestehender und neuer Packages, wächst die Leistungsfähigkeit der Sprache ständig (PYTHON 2024).

Conda: Bei Conda handelt es sich um ein Paketverwaltungssystem zur Einrichtung und Verwaltung von Python (oder R) Umgebungen. Die Versionierung der Programmiersprache sowie die installierten Bibliotheken können je Umgebung variieren. Dies ermöglicht die Konfiguration von Anwendungsfall bezogenen Umgebungen, die unabhängig voneinander agieren können. Hierbei ist die Berücksichtigung von Paketabhängigkeiten bei der Aktualisierung einzelner Softwarepakete hervorzuheben, die maßgeblich zur nachhaltigen Funktionalität der Anwendung beiträgt (ANACONDA, 2024).

PyCharm: Als Entwicklungsumgebung wurde in dieser Arbeit PyCharm in der Community Edition 2021.3.2 verwendet. PyCharm bietet neben klassischem Codeeditor mit intelligenter Codevervollständigung einen integrierten Debugger zur schrittweisen Fehlersuche innerhalb des Codes (PYCHARM 2024). Gerade bei umfangreichen Projekten mit mehreren vernetzten Skripten ist dies essenziell.

PyTorch Lightning: Bei PyTorch Lightning handelt es sich um die High-Level API auf Basis von PyTorch. Ziel ist hierbei der erleichterte Umgang mit PyTorch Code um die Entwicklung von ML/DL-Modellen zu vereinfachen. Vorteile von Lightning sind die Verwendung modularer Struktur (LightningModule, Trainer etc.) oder standardisierter Schnittstellen (vordefinierte Methoden für das Training (training_step, test_step oder validation_step) (LIGHTNING 2024). Standardisierung und wartungsfreundlicher Code sind speziell für komplexe Projekte unabdingbar.

Segmentation Models: Segmentation Models ist eine Python Bibliothek, die dem Nutzer Zugang zu vortrainierten Modellen und Tools für die Segmentierung von Bilddaten ermöglicht. Hier können gängige Netzarchitekturen (z.B. U-Net oder FPN) ausgewählt werden und je nach Belieben vortrainierte Gewichte verwendet werden. Weiterhin kann der Anwender für einen bestehende Architektur einen anderen Backbone bestimmen und dieses konfigurierte Netz neu trainieren (SEGMENTATION MODELS 2024). In dieser Arbeit wurden hier ein FPN mit ResNet50 Backbone gewählt.

NumPy: NumPy ist eine Bibliothek für numerische Berechnungen (**Num**erical **Py**thon). Speziell für große Datenstrukturen, wie mehrdimensionale Arrays und Matrizen erweist sich NumPy als leistungsstark, weswegen es oft Basis vieler weiterer Bibliotheken im Bereich der Data Science ist (NUMPY 2024). Aufgrund der Tatsache, dass Tensoren ebenfalls mehrdimensionale Vektoren sind, bietet NumPy auch für die Arbeit mit neuronalen Netzen eine solide Grundlage.

Geopandas: Geopandas fokussiert sich auf die Verarbeitung des Raumbezugs von Geodaten und basiert auf der Bibliothek Pandas. Pandas ist weit verbreitet im Bereich der Datenanalyse und -manipulation. Hierdurch lassen sich statistische Auswertungen (z.B. im Bereich der deskriptiven Statistik) machen, um ein Verständnis der Eingangsdaten zu erhalten. In dieser Arbeit ist GeoPandas unter anderem bei der Verarbeitung des LBMs und dessen Rasterisierung relevant.

Albumentations: Zur Verwendung von Augmentierungen kann die Bibliothek Albumentations genutzt werden. Hiermit lassen sich verschiedene Transformationen der Originaldaten vornehmen. Diese variieren von geometrischen oder Farbtransformationen bis hin zu Verzerrungen oder Pixel-Level-Transformationen (z.B. Bildrauschen) (ALBUMENTATIONS 2024). In dieser Arbeit wird sich auf geometrische Veränderungen beschränkt.

Seaborn und PyPlot: Bei PyPlot und Seaborn handelt es sich um Visualisierungsbibliotheken zur Erstellung von Graphen und Diagrammen. Diese können je nach Belieben in Aussehen und Beschriftung angepasst werden. In dieser Arbeit werden die Bibliotheken zu Erstellung des graphischen Outputs der Konfusionsmatrizen und der Klassenverteilungen nach Testgebiet verwendet.

GDAL: Die „**G**eospatial **D**ata **A**bstraction **L**ibrary“ ist eine Bibliothek zur Verarbeitung und Manipulation von geographischen Rasterdaten. Aufgrund grundlegender Operationen wie Transformationen von Datentypen, Koordinatensystemen oder Rasterformaten bildet GDAL die Voraussetzung für viele weitere Pakete (GDAL 2024). Unteranderem sind dies Geopandas, Fiona oder Rasterio.

Rasterio: Rasterio ist speziell auf die Arbeit mit Rasterdaten ausgerichtet. Die Bibliothek unterstützt zahlreiche Rasterformate, wie GeoTIFF, JPG oder PNG und zeichnet sich durch gute Integration mit NumPy aus. Dies ermöglicht eine schnelle Verarbeitung und Transformation der Daten.

Scikit learn: Scikit-learn ist eine weit verbreitete Machine-Learning-Bibliothek. Hier sind zahlreiche Algorithmen zur Regression, Klassifikation oder dem Clustering implementiert. Die Verwendung im DL-Kontext erfolgt häufig zur Vorverarbeitung der Daten oder der Dimensionsreduktion, sowie der Evaluation des Trainings. Neuronale Netze können hier lediglich ausgeführt, aber nicht trainiert werden (SCIKIT-LEARN 2024, DERU & NDIAYE 2019:45).

FME Form 2023.1: Die **F**eature **M**anipulation **E**ngine von Safe Software ist eine kommerzielle Software zur Verarbeitung und Manipulation von Daten aller Art. Mit über

1000 möglichen Datenquellen ermöglicht die FME die Bearbeitung und Ausgabe zahlreicher Formate (SAFE 2024). Mithilfe von hunderten Transformern können Benutzer ihre Eingangsdaten nach Belieben bearbeiten und beeinflussen. Im Bereich der Geodaten können sowohl Rasterdaten und Vektordaten als auch Punktwolken eingelesen werden. In dieser Arbeit wurde die FME verwendet, um die Konfusionsmatrizen bzgl. klassenspezifischer Metriken zu untersuchen und die Daten für die anschließende Visualisierung vorzubereiten. Weiterhin erfolgte die gesamte Vorbereitung der Planet Szenen, einschließlich der Umrechnung in TOA Reflektanzen, innerhalb der FME.

4 Auswertung und Diskussion

Dieses Kapitel beschreibt und diskutiert das erreichte Klassifizierungsergebnis, beginnend mit einer visuellen Evaluation auf Basis der Klassenspezifika der LBM-Klassen und deren Aktualisierung (4.1). Anschließend wird die Gesamtgüte des Verfahrens besprochen (4.2). Sowohl die visuelle Evaluation (4.1) als auch die Bewertung der Gesamtgüte (4.2) finden auf der Basis drei verschiedener Szenarien statt. Neben den 32 Klassen (31 Klassen des LBM + NoData; 1. Szenario) werden zwei Aggregierungsstufen mit 16 Klassen (Agg16, siehe Tabelle 8) und 9 Klassen (LBM-Metaklassen; Agg9) evaluiert (siehe Tabelle 8).

Tabelle 8: Beschreibung der drei betrachteten Szenarien zur Evaluation der Klassifizierungsergebnisse des eingesetzten Verfahrens für Schleswig-Holstein.

Szenario	Klassenanzahl	Beschreibung
1	32	Klassenspezifische Beurteilung gemäß den Klassen des LBM-DE. Zusätzlich wurde die Klasse 0 für NoData Werte hinzugefügt
2	16	Aggregierung auf 16 Klassen (Agg16). Neben den LBM-Klassen 1 bis 11 sind verschiedene Aggregierungen der restlichen Klassen vorgenommen worden. Die Klassen 12 und 13 des LBMs wurden hier in Klasse 12 aggregiert. Weiterhin wurde der Baumbestand (Klasse 14 bis 17 des LBMs) in Klasse 13, die Klassen 18 und 19 (LBM) in Klasse 14 sowie alle Wasserklassen (26 bis 31 des LBMs) in Klasse 15 aggregiert. Die Moorklassen (22 bis 25 des LBMs) wurden zur Klasse 9 hinzugefügt. Neben den NoData Werten beinhaltet Klasse 0 auch die Klasse Brandfläche und Schnee & Eis
3	9	Aggregierung auf 9 Klassen (Agg9). Die Klassen sind die LBM-Metaklassen + NoData (NoData (0), Bebauung (1), Ackerland (2), Grünland (3), Sträucher (B322 & B324; 4), Baumbestand (5), Vegetationslose Flächen (6), Feuchte Flächen (7) und Wasserflächen (8).

4.1 Evaluation des Klassifizierungsergebnisses

Die Einführung in den aktuellen Aktualisierungsprozess aus Kapitel 3.3 ist für die Interpretation des Klassifizierungsergebnisses dieser Arbeit relevant, speziell im Hinblick auf die dritte Leitfrage der Arbeit. Aufgrund der Aktualisierung bestehender Polygone, weiterer attributiver Unterteilung und Hinzunahme verschiedener Geodatenquellen (vgl. Kap. 3.3) sind der reinen semantischen Segmentierung auf Basis von Fernerkundungsdaten Grenzen gesetzt. Hierbei sind manche Klassen des LDM-DE nicht ohne weitere Postprozessierung auflösbar.

Der Vergleich der Klassifizierungsergebnisse der drei Szenarien (siehe Abbildung 15 bis Abbildung 17) zeigt eindrücklich, dass der Domänenwechsel von RapidEye zu PlanetScope durch das eingesetzte Verfahren möglich ist. Mit geringem Trainingsaufwand sind scharfe Abgrenzungen innerhalb der Landbedeckungstypen des LBM-DE möglich.

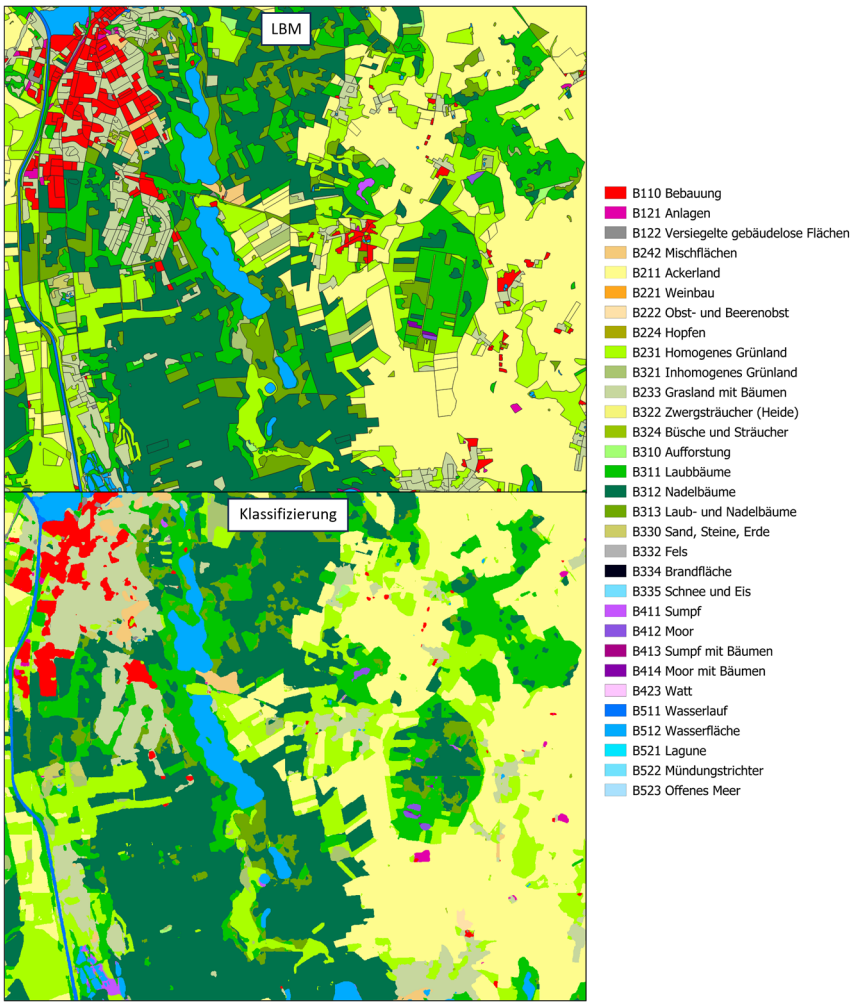


Abbildung 15: Evaluation der Klassifizierung für das erste Szenario auf Basis von 32 Klassen (1. Szenario). Vergleich der LBM-Labels (oben) mit dem Klassifizierungsergebnis (unten) am Beispiel eines Ausschnitts über Orbit 2430 (Eigene Darstellung).

Abbildung 15 zeigt dies eindrücklich im Vergleich zwischen den Labels des LBM und der Klassifizierung des Verfahrens. Sowohl urbanere Bereiche als Kombination von B110 („Bebauung“) und B233 („Grasland mit Bäumen“), höhere Vegetation (B31x), Grünland (B231) als auch Ackerland (B211) und die Wasserflächen (B5xx) lassen sich deutlich voneinander abgrenzen. Die Aggregierungen von Szenario 2 und 3 (Agg16 und Agg9) erleichtern ebenfalls die Klassifizierung, da die Klassen innerhalb einer Landbedeckungskategorie häufig spektral sehr ähnlich sind, was die Abgrenzung erschwert. Es zeigt sich das intrakategoriale Abgrenzungen wie beispielsweise bei der höheren Vegetation (B31x) schwierig sind. Vergleicht man die Auswertungen der drei Szenarien fällt auf, dass die Abgrenzung zwischen den verschiedenen Vegetationsklassen B311 „Laubbäume“, B312 „Nadelbäume“ und B313 „Laub- und Nadelbäume“ schwierig ist.

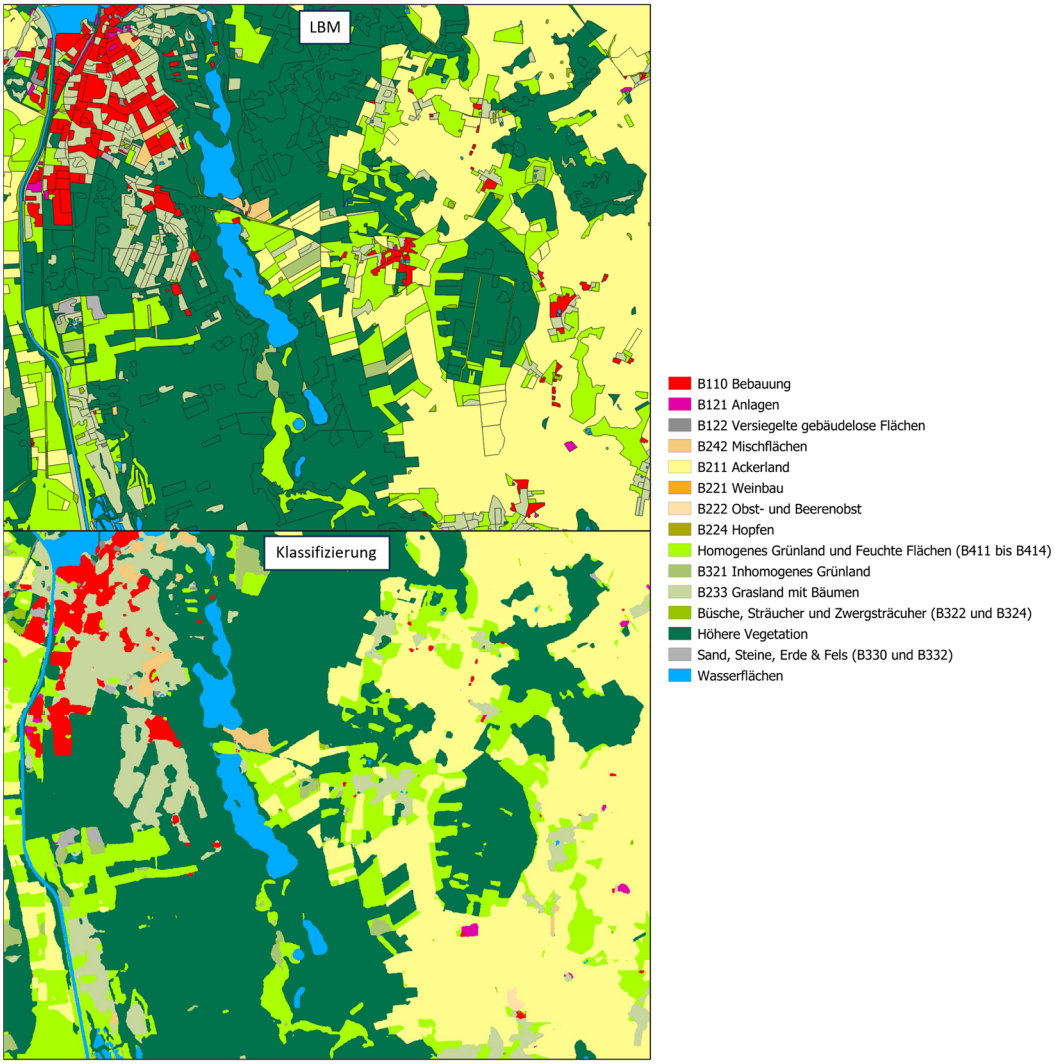


Abbildung 16: Evaluation der Klassifizierung für das zweite Szenario auf Basis von 16 Klassen (2. Szenario). Vergleich der LBM-Labels (oben) mit dem Klassifizierungsergebnis (unten) am Beispiel eines Ausschnitts über Orbit 2430 (Eigene Darstellung).

Während sich das Modell in Szenario 1 (Abbildung 15) an manchen Stellen unsicher ist, führen die beiden Aggregierungen von Szenario 2 (Abbildung 16) und Szenario 3 (Abbildung 17) zur Eliminierung dieser Unsicherheiten. Dies muss nicht für eine unzureichende Qualität des Modells sprechen, sondern kann aus einem Zusammenspiel veränderter Ist-Zustände innerhalb des Waldes und schwierig modellierbarer Klassenbeschreibungen (z.B. B313 Laub- und Nadelbäume) resultieren. Näheres hierzu im weiteren Verlauf des Kapitels.

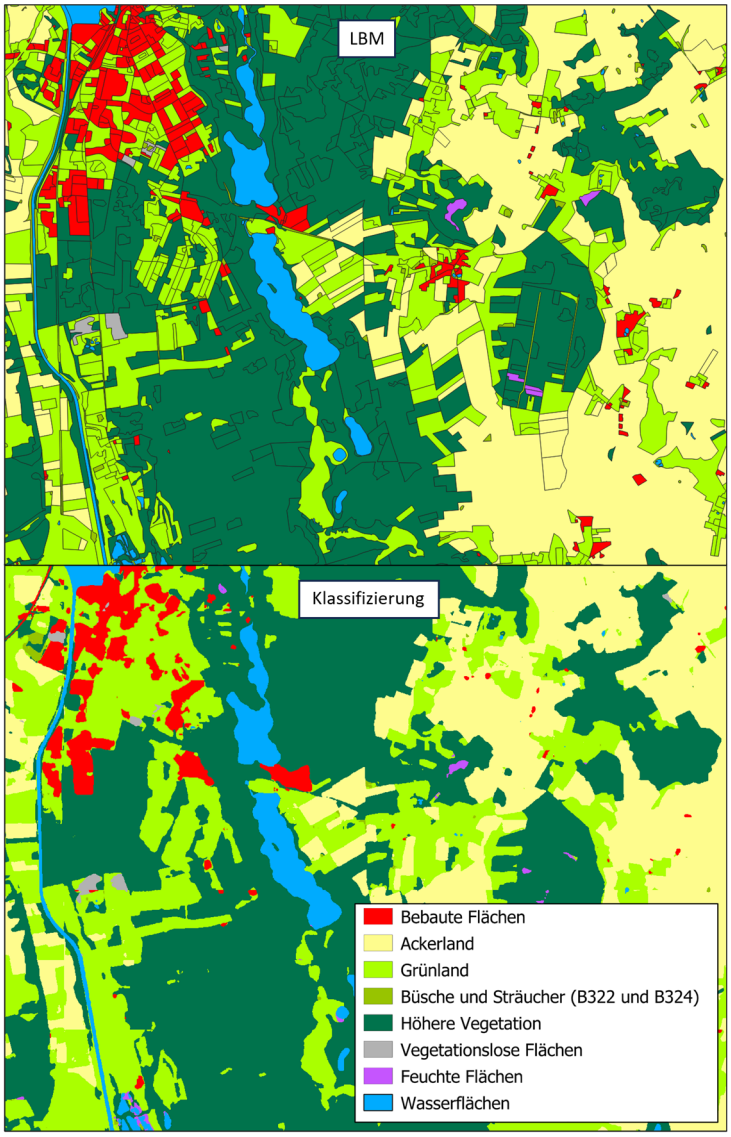


Abbildung 17: Evaluation der Klassifizierung für das dritte Szenario auf Basis von 9 Klassen (3. Szenario). Vergleich der LBM-Labels (oben) mit dem Klassifizierungsergebnis (unten) am Beispiel eines Ausschnitts über Orbit 2430 (Eigene Darstellung).

Diese und weitere Herausforderungen in der Interpretation der Modellvorhersagen sollen nun im weiteren Verlauf des Kapitels näher beleuchtet werden. Zur Evaluation der Herausforderungen der semantischen Segmentierung folgen nun einige Abbildungen, die die wesentlichen Charakteristika der Klassifizierung zusammenfassen.

Als erste Eigenschaft zeichnet das Modell „Intrakategoriale Homogenität“ aus. Hier ist sich das Modell zum einen aufgrund der Eingangsdaten, zum anderen aufgrund der LB-Klassendefinition unsicher, welche Klasse die Richtige ist. Abb. 18 zeigt in der ersten Zeile Unsicherheiten bei der Klassendefinition. Speziell geht es hier um die Klassen B311 (Laubbäume), B312 (Nadelbäume) und B313 (Laub- und Nadelbäume). Das Modell hat Probleme die Mischklasse B313 zuzuweisen und tendiert zur Separierung von Laub- und Nadelwäldern. Vergleicht man die LBM-Klassen (Abb. 18 (b)) mit der Vorhersage (Abb. 18 (c)) ist zu erkennen, dass die Mischklasse kaum ausgewiesen wird. Die vorhandenen Mischwälder werden vom Modell als Laubwälder mit einzelnen Bereichen von Nadelwäldern innerhalb der gesamten Waldstruktur erkannt. Wobei der reine Laubwald und reine Nadelwald recht gut getroffen werden. Diese Eigenschaft setzt sich durch alle Orbits durch. Die Mischwaldklasse B313 wäre eine klassische Postprozessierungsklasse. Die festgesetzten Polygone auf denen aktuell das LBM aktualisiert wird, tragen hier maßgeblich zur Klasse bei. Schaut man in die Beschreibung der Klasse (siehe Anhang 10) handelt es sich bei B313 um „Flächen, die zu mindestens 50% mit Bäumen bestanden sind. Keine Waldart darf mehr als 75% dieser Fläche ausmachen. Es muss eine baumweise oder baumgruppenweise Durchmischung von Laub- und Nadelbäumen erkennbar sein.“ (BKG 2023). Hierfür müssen vorher Flächen definiert, oder nachträglich ausgewiesen werden, um solche Zuweisung zu betreiben. Zusätzlich wirkt der PlanetScope Ausschnitt (Abb. 18 (a)) für dieses Beispiel recht homogen, bzw. sind die visuellen Unterschiede zwischen B313 und B311 des LBMs (b) recht gering, was eine Zuweisung zur selben Klasse begünstigt.

Als weiteren Prozessierungsschritt könnte das Klassifizierungsergebnis genutzt werden, um die prozentuale Verteilung innerhalb der Flächen zu überprüfen. Eine reine Ausweisung von Klasse B313 ohne weitere Verarbeitung ist hier nur schwer möglich. Der nächste Fall (Abb. 18 (d-f)) bezieht sich auf die beiden häufigsten Klassen B231 (Homogenes Grünland) und B211 (Ackerland). Zum einen lässt sich im Vergleich der Abbildungen d bis f (Abb. 18) erkennen, dass Ackerflächen je nach Bewuchs Ähnlichkeiten mit Grünlandflächen aufweisen können

Abbildung 18: Visuelle Evaluation der Klassifizierungsergebnisse. Verglichen werden PlanetScope Ausschnitte (1.Spalte) mit den Labels des LBM (2. Spalte) und den Vorhersagen des Modells (3.Spalte). Zeilenweise sind verschiedene Herausforderungen bei der pixelweisen Segmentierung in Bezug auf das LBM-DE dargestellt. Die erste Zeile zeigt Einschränkungen bei der Zuweisung von Mischklassen (a-c) und die zweite Zeile (d-f) konzentriert sich auf B231 und B211. Die dritte Zeile bezieht sich auf die erschwerte Differenzierung von B110 und B233 (Eigene Darstellung).

Da in dieser Arbeit Satellitenszenen aus dem Monat Juni verwendet werden, befinden sich viele Pflanzen im Wachstum, was je nach Zeitpunkt der Bestellung des Ackers unterschiedlich aussehen kann. Betrachtet man die Bilder d bis f in der Mitte rechts, lässt sich ein Bereich identifizieren, wo die Labels des LBMs Unterschiede zwischen Homogenem Grünland und Ackerland vorhersagen. Der Planet Ausschnitt (Abb. 18 (d)) zeigt allerdings ähnliche Bewirtschaftung. Gerade bei Ackerland und Grünland kommt es häufiger vor, dass sich eine Landbedeckung auf Kosten der anderen ausbreitet und dadurch die Labels nicht immer korrekt sind. Dies kann jedoch jährlich variieren, wodurch eine manuelle und stetige Anpassung des Datensatzes und der dazugehörigen Labels erschwert wird. Eine weitere Besonderheit bei der Klasse „Homogenes Grünland (B231)“ ist deren Definition. Hierbei handelt es sich um Flächen „mit durchgehendem Grasbestand, die regelmäßig beweidet oder gemäht werden“ (BKG, 2023). Um eine Regelmäßigkeit einer Beweidung oder Mahd zu erkennen, benötigt man eine höhere temporale Auflösung. Häufig werden in solchen Fällen weitere Satellitendaten mit hoher temporaler Auflösung und einfachem Zugang verwendet, um Veränderung innerhalb eines Jahresgangs zu untersuchen (z.B. Sentinel-2). Ähnliche Einschränkungen gelten auch für die anderen beiden Klassen der Kategorie Grünland, B321 „Inhomogenes Grünland“ und B233 „Grasland mit Bäumen“. Bei B321 geht es um „Grünlandflächen, die höchstens einmal jährlich bearbeitet werden. Unregelmäßiges Erscheinungsbild, häufig mit Stauden und Gestrüpp durchsetzt.“ (BKG 2023). Häufig werden diese Flächen aus selbigen Gründen nicht gut erkannt, da das Modell lediglich die Beschaffenheit der Daten zu einem einzigen Zeitpunkt untersuchen kann. Dies führt öfter zu Fehlzuweisungen zu den Grünlandklassen aus den Bereichen niedriger Vegetation (z.B. B322 „Zwergsträucher (Heide), B324 „Büscher und Sträucher“ und B310 „Aufforstung“). Zur besseren Unterscheidung kann ein aktuelles normalisiertes digitales Oberflächenmodell (nDOM) beitragen, welches Höhen der Objekte auf der Erdoberfläche anzeigt. Hierüber könnte man bei hoher Aktualität eine Differenzierung von niedriger und hoher Vegetation vornehmen.

Die Dominanz der Grünlandklassen, hier speziell B233 „Grasland mit Bäumen“, zeigt sich auch in Kategorie der bebauten Flächen (B110 „Bebauung“, B121 „Anlagen“, B122 „Versiegelte gebäudelose Flächen“ und B242 „Mischflächen“). Abb. 18 (g-i) zeigt größere Unsicherheiten in der Unterscheidung von B233 und B110. Begrünte Siedlungsstrukturen, wie grünere Standrandbereiche oder kleinere Orte, werden aufgrund des ho-

hen Grünungsanteils häufig als B233 klassifiziert (vgl. Abb. 18 (i)). Nimmt die Begrünung ab (vgl. Abb. 18 (i) Mitte und unterer Bildrand), erfolgt die korrekte Zuweisung von B110. Eine Unterscheidung innerhalb der bebauten Flächen bei der Klassifizierung gelingt dem Modell häufig weniger gut, speziell wenn einzelne Flächen komplett von einer anderen LB eingeschlossen sind. Als Beispiel dienen hier die Klassen B121 und B110 in Abbildung 18 (i). Hier werden einzelne Flächen der B121 entweder zur B110 oder B233 zugeordnet. Näheres zu Einschränkungen mit kleineren, vereinzelten Flächen folgt später innerhalb dieses Kapitels.

Ein weiteres Beispiel für die Schwierigkeit der intrakategorialen Differenzierung liefern die Wasserklassen. Schaut man sich die Klassen innerhalb von Kategorie G (Wasserflächen) an, fällt auf, dass abseits des offenen Meeres (B523) die intrakategoriale Differenzierung schwerfällt. Ein visuelles Beispiel liefert Abb. 19 (a-c). Während B512 („Wasserfläche“) im oberen Teil des Ausschnitts (Abb. 19 (c)) noch gut erkannt wird, hat das Modell in der Mitte und den südöstlichen Regionen des Bildes Probleme. Hier variiert man zwischen B511 („Wasserlauf“), B512 („Wasserfläche“), B521 („Lagune“), B522 („Mündungstrichter“), B523 („Offenes Meer“) und B423 („Watt“). Die Küstenregionen unterliegen durch die Gezeiten größeren Fluktuationen im Wasserstand. Das macht hier die Differenzierung in einzelne Landbedeckungsklassen, auf der Basis einer PlanetScope Szene, zu einem einzigen Zeitpunkt besonders schwer. In der Mitte des Planet Ausschnitts (Abb. 19 (a)) erkennt man hier Unterschiede im Wasserstand zwischen küstennahen und küstenferneren Regionen, wo das Modell in Küstennähe B423 auswies (Abb. 19 (c)). Ebenfalls wurden schmale Verästelungen der Klasse B523 innerhalb des ausgedehnten Wattbereichs (Abb. 19 (b), westlich im Bild) nicht erkannt und der Klasse B423 zugewiesen (c). Beim Blick in die Klassenbeschreibungen (Anhang 10) der Wasserklassen fallen Beschreibungen wie „zwischen den Niveaus des mittleren Hoch- und mittleren Niedrigwasserstands“ (B423) oder „der an den mittleren Niedrigwasserstand angrenzende Bereich“ (B523) auf. Hierfür sind höhere temporale Auflösungen notwendig, sodass das Modell in der Lage ist die Dynamik der Wasserstände für spezielle Regionen abzuleiten. Weiterhin sind anschließende Prozessierungen zur räumlichen Differenzierung vorstellbar. Speziell für Klassen B511, B512, B521 und B522 gibt die Beschreibung vor, welchen räumlichen Einschränkungen diese Flächen unterlegen sind. Bspw. handelt es sich bei der LB Klasse B521 („Lagune“) um „Salz- oder Brackwasserzonen im Küstenbereich, die vom Meer durch eine Landzunge o.ä. getrennt sind“ (BKG, 2023).

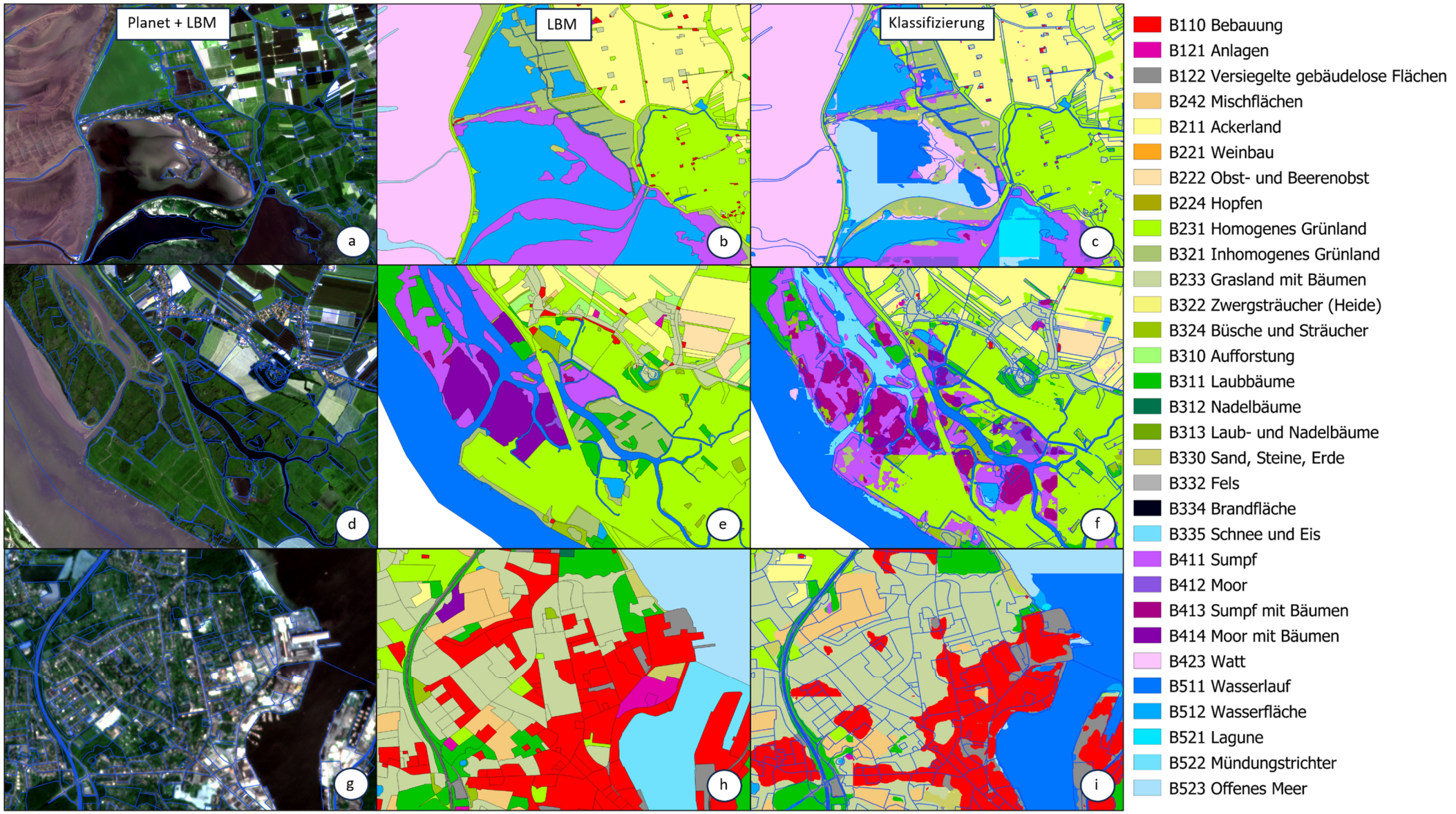


Abbildung 19: Visuelle Evaluation der Klassifizierungsergebnisse. Verglichen werden PlanetScope Ausschnitte (1.Spalte) mit den Labels des LBM (2. Spalte) und den Vorhersagen des Modells (3.Spalte). Zeilenweise sind verschiedene Herausforderungen bei der pixelweisen Segmentierung in Bezug auf das LBM-DE dargestellt. Die erste Zeile zeigt Unsicherheiten bei der intrakategorialen Unterscheidung, hier am Beispiel der Wasserklassen (a-c) und die zweite Zeile (d-f) konzentriert sich auf die Unterscheidung der feuchten Flächen (B41x). Die dritte Zeile gibt weitere Eindrücke zur Differenzierung von B110 und B233, sowie der Unsicherheiten in Bezug auf die Wasserflächen (Eigene Darstellung).

So könnte man im Nachgang zur Klassifizierung noch spezielle Regionen ausweisen, wo vereinzelte Klassen auftreten können, um die Qualität der Klassifikation zu steigern.
Als weitere Landbedeckungskategorie ist die Klassifizierung der feuchten Flächen aufgrund deren Eigenschaften problematisch. Die feuchten Flächen setzen sich aus Sümpfen (B411) und Mooren (B412) und selbigen mit Bewuchs (B413, „Sumpf mit Bäumen“ und B414, „Moor mit Bäumen“) zusammen. Sowohl B411 als auch B412 sind unbewaldete, nasse Flächen mit dem Unterschied, dass Moore (B412) aus Torfmoos und unvollständig abgebauten pflanzlichen Stoffen bestehen müssen (BKG, 2023). Liegt zusätzlich eine Busch- und Baumvegetation vor, die 30-50% der bestehenden Fläche einnimmt, weist man die Klassen B413 und B414 aus. Bei höheren Anteilen weicht man auf B31x Klassen aus. In der Realität sind die feuchten Flächen häufig nicht gut untereinander zu unterscheiden. Weiterhin ähneln sie ebenfalls Grünlandklassen oder anderer Vegetation und kommen in SH nur vereinzelt vor, wodurch sich das Verfahren hier sehr unsicher ist. Abb. 19 (d-f) bestätigt diese Unsicherheit. Während der PlanetScope Ausschnitt (d) recht homogen aussieht, treten hier eine Vielzahl unterschiedlichster Klassen auf. So setzt sich die Mitte des Ausschnitts aus B411, B413 und B414 zusammen und geht dann östlich in ein Konglomerat aus verschiedenen Grünlandklassen sowie B311 („Laubbäume“) über (siehe Abb. 19 (e)). Diese Zuweisung gelingt dem Modell mit vereinzelten räumlichen Ausnahmen bei B411 weniger. Interessant sind hier auch die Labels des LBM, wodurch die Mitte des Ausschnitts primär von Klasse B411 und B414 eingenommen wird. Instinktiv hätte man hier B411 und B413 angenommen. Auf alleiniger Basis der Planet Szene ist hier nur schwerlich eine räumliche Differenzierung möglich, da diese recht homogen wird. Nützlich wären hier detailliertere Informationen zur Feuchte und der Zusammensetzung des Bodens.
Die letzte Zeile (g-i) von Abb. 19 fasst einige erwähnte Charakteristika des Modells nochmal zusammen. Im östlichen Bereich des Bildes zeigt das Modell Schwierigkeiten bei der Zuweisung der genauen Wasserklasse (intrakategoriale Homogenität). Weiterhin gibt es Schwierigkeiten bei der Unterscheidung verschiedener Bebauungsklassen sowie der Erkennung vereinzelter Flächen, die von einer anderen LB umschlossen sind. Das Modell tendiert zusätzlich zur Ausweisung von B233 („Grasland mit Bäumen“) in grüneren Stadtteilen. Zuletzt ist es bereits in allen Bildern aufgefallen, dass das Modell Schwierigkeiten hat, die genaue Polygongrenze als Wechsel zwischen zwei unterschiedlichen Landbedeckungen zu treffen. Solche Bereiche sind von sog. Mischpixeln geprägt, die

eine klare Abgrenzung voneinander erschweren. Bei der Digitalisierung der Ursprungsflächen wurden Polygongrenzen manuell von Bearbeitern erfasst, die nach deren Auffassung die Unterscheidung verschiedener LBs vornahmen.

Diese gesetzten Grenzen können je nach Auffassung des Bearbeiters sowie mit variierender räumlicher Auflösung zu Ungenauigkeiten führen. Zusätzlich gibt es häufig keine scharfen Kanten bei verschiedenen Landbedeckungsklassen. Ein eindrückliches Beispiel liefert Abb. 20. Hier sind die Vorkommen der Klasse B324 („Büsche und Sträucher“) innerhalb einer Agrarlandschaft dargestellt. Die dominanten Klassen sind hier B211 („Ackerland“) und B231 („Homogenes Grünland“). Die LBM-Labels in (b) sind hier transparent mit Ausnahme der Klasse B324 dargestellt, um die kleinen Flächen dieser Klasse hervorzuheben. In der Klassifizierung fällt auf, dass viele der kleinen Flächen entweder in Teilen oder komplett einer anderen Landbedeckung zugewiesen wurden. Das gleiche Phänomen erkennt man auch bei der Klasse B110 („Bebauung“) in der Mitte des Bildes, wo lediglich die Kerne der Polygone der richtigen Klasse zugewiesen wurden. Diese Beispiele unterstreichen die erschwerte Zuweisung im Randpixel/Mischpixelbereich von Landbedeckungsklassen, besonders, wenn eine kleine isolierte Landbedeckungsfläche von einer anderen Landbedeckung umschlossen ist. Einen Überblick über das Verhältnis von Vorkommen in Bezug auf Fläche pro Landbedeckung liefert der Vergleich von Abb. 14 (Fläche pro Landbedeckung in %) und Anhang 11 (Anzahl Objekte pro Landbedeckung).

Neben den zahlreichen diskutierten Charakteristika des Modells ist die Qualität der Eingangsdaten sowie richtig zugewiesener Labels für den Trainingserfolg entscheidend. Durch eine intensiv genutzte Landoberfläche bilden die Labels nicht immer den aktuellen Ist-Zustand ab. Häufig sind dies Klassen, die bspw. in der Nutzung geändert wurden, wie Ackerland und Grünland (siehe Abb. 20, Mitte). Zusätzlich betrifft dies, aber auch weitere Klassen, wie die höhere Vegetation, die durch die Klimaextreme der letzten Jahre gelitten hat und somit nun häufiger B310 („Aufforstung“) Flächen entstehen. In diesem Zusammenhang sind ebenfalls die Aufnahmezeitpunkte der Satellitendaten im Verhältnis zur Aktualisierungsperiode des LBMs (drei Jahre vor Bezugsjahr der Ausgabe) zu setzen.

Abbildung 20: Visuelle Evaluation der Klassifizierungsergebnisse. Verglichen wird ein PlanetScope Ausschnitt (a) mit den Labels des LBM (b) und der Vorhersage des Modells (c). Dargestellt ist die Schwierigkeit der Erfassung kleiner, alleinstehender Flächen am Beispiel der Klasse B324 („Büsche und Sträucher“). Vor diesem Hintergrund wurden alle LBM-Klassen abseits von B324 in deren Deckkraft reduziert (b) (Eigene Darstellung).

4.2 Bewertung der Gesamtgüte

Nach der visuellen Evaluation des Klassifizierungsergebnisses sollen getroffene Aussagen anhand verschiedener Gütemetriken gefestigt werden. Im Rahmen der pixelweisen semantischen Segmentierung werden hierzu häufig Konfusionsmatrizen verwendet. In diesen Matrizen werden die eigentlichen Klassen (aufgetragen auf der Ordinate) mit den vorhergesagten Klassen (Abszisse) verglichen. In den einzelnen Zellen der Matrix wird die Anzahl der Pixel notiert, welche diesen Klassen zugewiesen wurden. Würde das Modell optimale Vorhersagen treffen, wäre lediglich die Diagonale der Matrix mit Werten

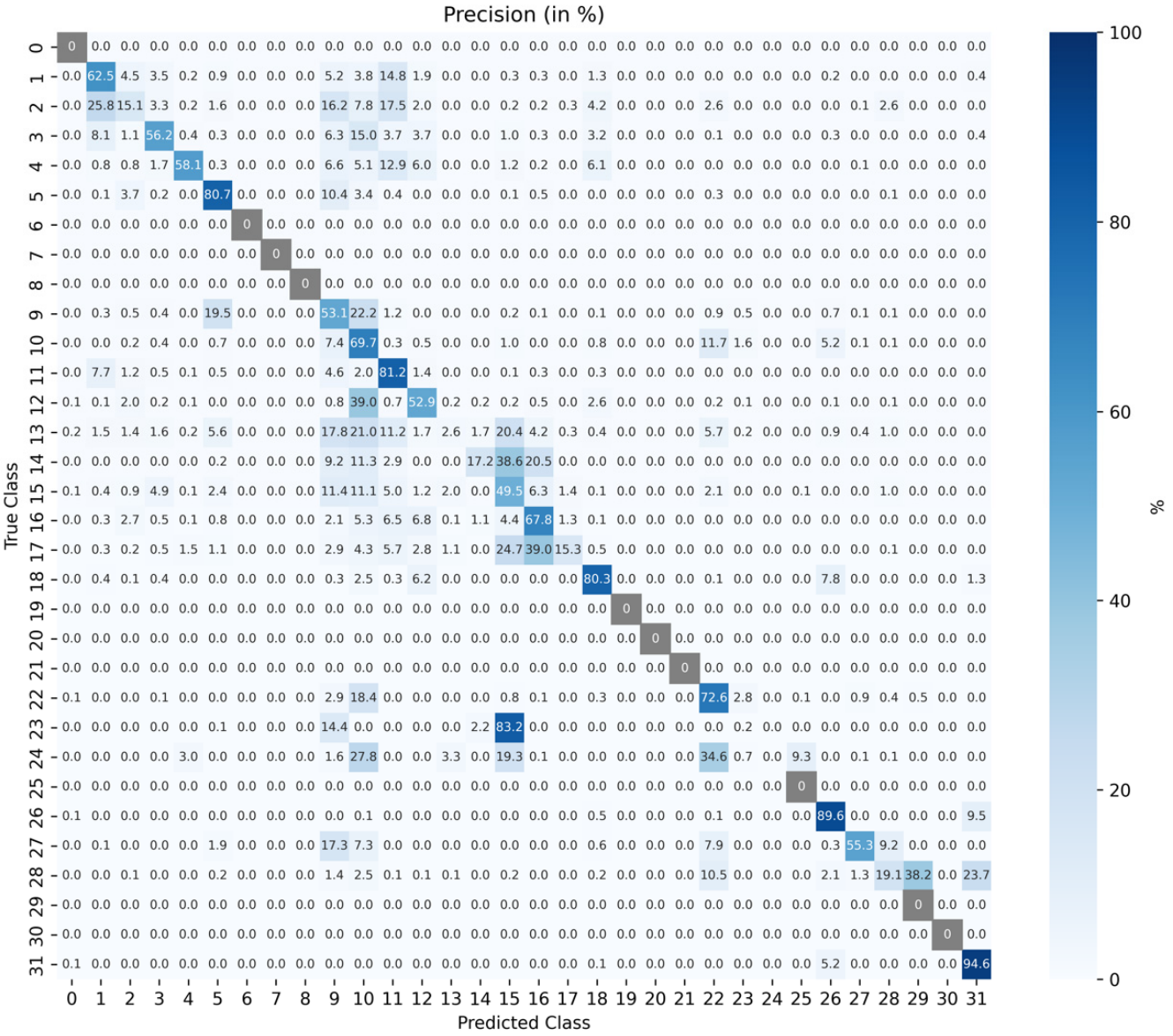


Abbildung 21: Konfusionsmatrix für die Präzision auf Basis des Testorbits 245c. Aufgetragen sind wahren Klassen (True Class) und die vorhergesagten Klassen (Predicted Class) für die 31 Landbedeckungsklassen des LBM-DE (und NoData als Klasse 0). Die Präzision ist in Prozent dargestellt, wobei graue Felder auf nicht vorhandene Klassen (oder NoData) hinweisen (Eigene Darstellung).

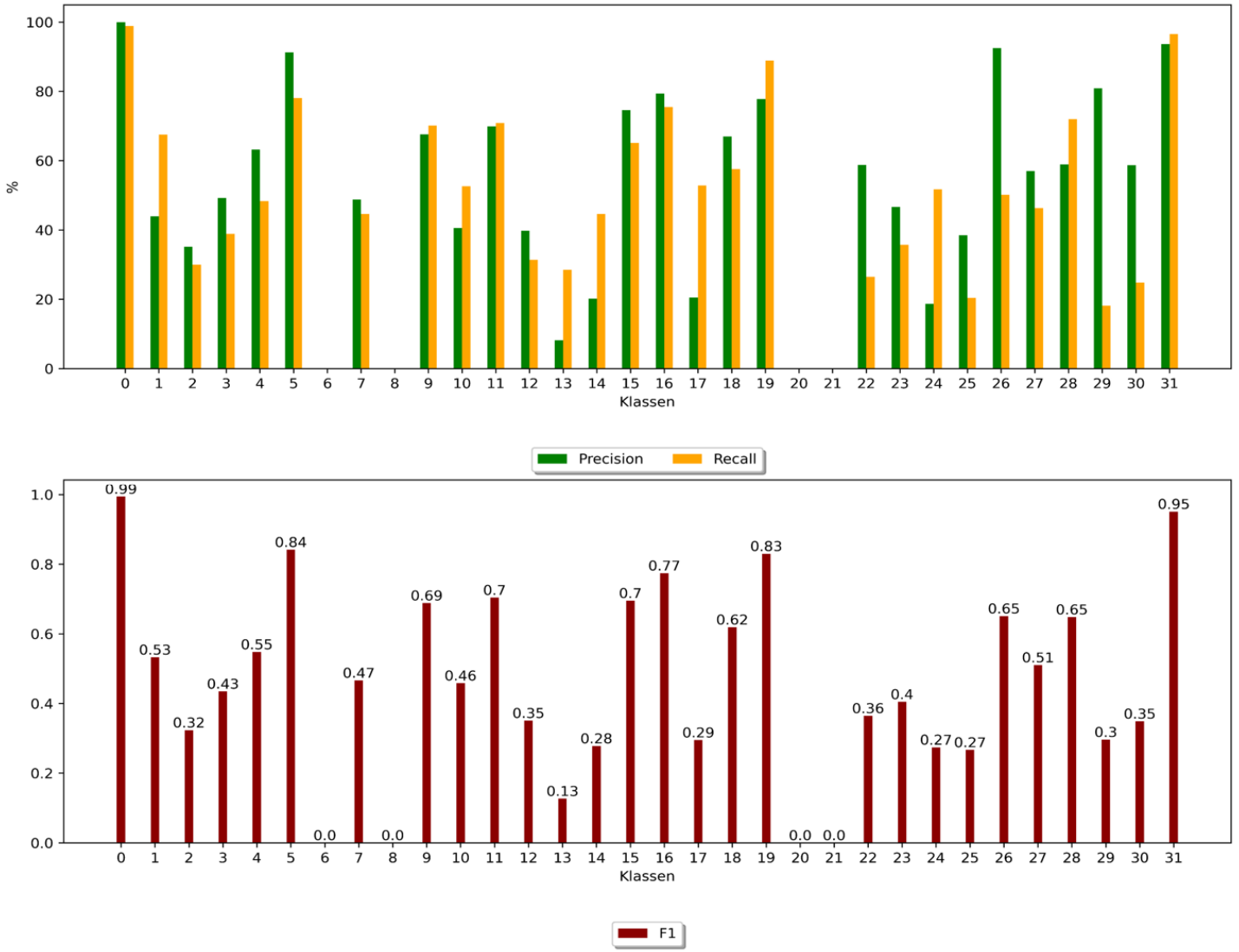
gefüllt. Zum besseren Verständnis wurde ein solche Matrix am Beispiel der Präzision für den Orbit 245c visualisiert (siehe Abb. 21).

Auf Basis dieser Matrix kann nun evaluiert werden, welche Pixel der richtigen Klasse (TP = True Positive), bzw. richtigerweise nicht dieser Klasse zugewiesen wurden (TN = True Negative). Gleichermaßen geben sie Auskunft, welche Pixel fälschlicherweise der jeweiligen Klassen zugewiesen (FP = False Positive) oder eben nicht zugewiesen wurden (FN = False Negative). Bei der Präzision geht es nun darum das Verhältnis der tatsächlich positiven Fälle aus der Gesamtzahl der als positiv klassifizierten Fälle zu ermitteln. In Abb. 21 ist die Präzision in Prozent für alle 32 Klassen angegeben, wobei die grauen Kästen Klassen zeigen, die im Orbit nicht enthalten sind. In der Diagonalen sind die Übereinstimmungen von eigentlicher Klasse (True Class) zur vorhergesagten Klasse (Predicted Class) aufgetragen. So ist zu erkennen, dass beispielsweise bei Klasse 11 81,2% der positiven Instanzen richtigerweise auch Klasse 11 zugeschrieben wurden. Die anderen 18,8% sind als falsch positive Instanzen (FP) anderen Klassen zugewiesen worden. Bei der Präzision wird zeilenweise gedacht, da sich hier die Prozente pro Zeile auf 100% aufsummieren lassen. Beim Recall wiederrum wäre es spaltenweise, da hierbei die richtig vorhergesagten Positiven zur Gesamtzahl der positiven Fälle pro Klasse betrachtet werden. Hier würden demnach die Spalten 100% in Summe ergeben. Ein beispielhafte Konfusionsmatrix für den Recall desselben Orbits kann im Anhang eingesehen werden (Anhang 4).

Präzision und Recall sind zwei Metriken, die eng zusammenspielen und in Kombination einen Eindruck zur Qualität des Klassifizierungsergebnisses geben. Hohe Präzisionswerte signalisieren, dass das Modell nicht viele falsche positive Vorhersagen (FP) macht. Jedoch könnte dies auch durch Modelle erlangt werden, die generell nicht viele positive Vorhersagen machen, was in einem niedrigen Recall resultiert (TERVEN ET AL. 2023, POWERS 2020). Bei guter Modellperformance sollten demnach sowohl Präzision als auch Recall hohe Werte einnehmen. Dadurch ist gewährleistet, dass die positiven Vorhersagen auch tatsächlich positiv (TP) sind (Präzision), und der Großteil an TP im Datensatz erkannt wird (Recall). Der F1 Score als harmonisiertes Mittel zwischen Präzision und Recall bewertet diese beiden Fähigkeiten eines Modells. Hohe F1 Scores sprechen daher für eine gute Balance zwischen Präzision und Recall. Ist eine der beiden Metriken ausgeprägt, spricht dies für niedrige F1 Scores.

Als Ergebnis der Kreuzvalidierung über alle Orbits zeigt Abbildung 22 die gemittelten klassenspezifischen Metriken. Auffällig sind einige Klassen mit hohen F1 Scores >= 0,70. Explizit Klasse 31 (B523 „Offenes Meer“) mit F1 = 0,95 und Klasse 5 (B211 „Ackerland“) mit F1 = 0,84 sind hier hervorzuheben. Weiterhin ist in Abbildung 22 zu erkennen, dass einige Klassen (B121: Anlagen, B122: Versiegelte gebäudelose Flächen, B321: Inhomogenes Grünland, B324: Büsche und Sträucher, B310: Aufforstung, B313: Nadel- und Laubbäume, sowie einzelne Wasserflächen (B521 und B522) und feuchte Flächen (B411 bis B414)) durch niedrigere F1 Scores (unter 0,5) auffallen (Abbildung 22). Der F1 Score als harmonisiertes Mittel zwischen Präzision und Recall bestätigt die ermittelten Charakteristika aus Kapitel 4.1. So lassen sich die niedrigen F1 Scores der feuchten Flächen (unter 0,40) mit der bereits erwähnten Schwierigkeit der Klassifizierung dieser Klassen in Verbindung bringen. Weiterhin sieht man im Bereich der Wasserklassen, dass die Klassifizierung abseits von B523 mit F1 = 0,95 erschwert ist. Hier fallen besonders B521 (F1 = 0,3) und B522 (F1 = 0,35) auf. Im Bereich der höheren Vegetation weisen die Klassen B311 (Laubbäume) und B312 (Nadelbäume) F1 Scores über 70% auf, wohingegen die Mischwaldklasse mit 0,29 schwierig zuzuweisen ist. Mögliche Ursachen wurden bereits in Kap. 4.1 erwähnt. Für die Klasse der Büsche und Sträucher (B324) wurden die niedrigsten F1 Scores mit 0,13 erzielt. Die bebauten Klassen (B110, B121, B122 und B242) weisen ebenfalls niedrigere Scores (F1 unter 0,55) auf.

Häufig lassen sich solche Unterschiede in den F1 Scores mit der Zusammensetzung der Landbedeckung des Untersuchungsgebiets erklären. Bei Klasse B211 handelt es sich mit 34,85% der Gesamtfläche von Schleswig-Holstein (Abbildung 14) um die häufigste Klasse in SH. Dem Verfahren werden hier eine größere Menge an Trainingsdaten der Landbedeckung zur Verfügung gestellt, wodurch die klassenspezifischen Charakteristika besser erfasst und gelernt werden können. Das Resultat sind häufig höhere F1 Scores, wie hier bei B211 mit F1 = 0,84. Gerade auch im Hinblick der Gesamtgenauigkeit (Korrekt klassifizierte Pixel im Verhältnis zu den gesamten Pixeln (TAYLOR 1996)) ist die Häufigkeit der vorkommenden Klassen im Untersuchungsgebiet entscheidend. Vergleicht man die Klassen mit den niedrigen F1 Scores mit deren Vorkommen in SH, macht keine dieser Klassen mehr als 2% der Landesfläche aus. Hier muss bedacht werden, dass einige der Klassen des LBMs sehr heterogen sind, wodurch das Modell bei wenig Daten pro Klasse nicht in der Lage ist, deren Charakteristika zu erfassen und anschließend zu detektieren (siehe Kap. 4.1).



Klasse (M)	Klasse (LBM)
0	
1	B110
2	B121
3	B122
4	B242
5	B211
6	B221
7	B222
8	B224
9	B231
10	B321
11	B233
12	B322
13	B324
14	B310
15	B311
16	B312
17	B313
18	B330
19	B332
20	B334
21	B335
22	B411
23	B412
24	B413
25	B414
26	B423
27	B511
28	B512
29	B521
30	B522
31	B523

Abbildung 22: Klassenspezifische Gütemetriken für die 31 Klassen des LBM + NoData (0) (1. Szenario). Aufgetragen sind die gemittelten Werte für Präzision und Recall (oben) sowie F1 (unten) für die Kreuzvalidierung über alle Orbits. Die Klassenzuweisung zwischen Modell (M) und LBM ist in der Tabelle (rechts) zu entnehmen. Die Klassen 6, 8, 20, und 21 sind im Untersuchungsgebiet nicht vorhanden (Eigene Darstellung).

Zusätzlich sind die Klassen innerhalb der einzelnen Oberkategorien häufig sehr ähnlich und spektral sowie visuell schwierig voneinander zu unterscheiden. Die Konfusionsmatrix in Abb. 21 zeigt solche intrakategorialen Ähnlichkeiten für Grünland (B231, B321 und B233) oder höhere Vegetation (B324 sowie B310 bis B313). Ähnliche Muster lassen sich in anderen Orbits auch für die anderen Kategorien erkennen.

Um die Modellgüte ohne diese Unsicherheiten zu evaluieren, wurden verschiedene Aggregierungsstufen der Szenarien 2 und 3 (siehe Tabelle 8) gebildet. Die Modellgüte wurde in diesem Zusammenhang einmal auf 16 Klassen (Agg16, Abb. 23) und einmal auf 9 Klassen (Agg9, Abb. 24) evaluiert. Die Klassen 1 bis 11 in Agg16 (Szenario 2) sind gleich geblieben im Vergleich zum Modell mit 32 Klassen (Szenario 1). Bei der höheren Vegetation wird zwischen Baum- und Strauchstrukturen unterschieden. Hierbei sei angemerkt, dass bei der Ableitung des LBMs dem Bearbeiter ebenfalls ein normalisiertes digitales Oberflächenmodell (nDOM) zur Verfügung steht. Dieses steht dem Modell in dieser Arbeit nicht zur Verfügung, was die Erkennung von essenziellen Höhenunterschieden innerhalb der Vegetationsklassen unmöglich macht. Jedoch unterscheiden sich Sträucher und Bäume häufig in Form und Lage. Die letzten beiden Kategorien (vegetationslose Flächen und Wasserflächen) wurden pro Kategorie zusammengefasst und nicht vorhandene Klassen als NoData gesetzt. Ebenfalls wurden die feuchten Flächen (B411 bis B414) zu homogenem Grünland zugeführt. Die Aggregierung zu 16 Klassen hat in den aggregierten Klassen zur Steigerung der F1 Scores beigetragen (Abbildung 23).

Speziell die Klassen 13 bis 15 haben von der Aggregierung profitiert. Bei 32 Klassen hatte das Modell Schwierigkeiten die Klasse B313 Nadel- und Laubwälder zu erkennen, da diese schwer zu differenzieren sind, von den reinen Laub- (B311) und Nadelwäldern (B312). Durch Zusammenführen dieser Klassen und Aufforstung (B310) sind alle Baumstrukturen innerhalb einer Klasse angesiedelt (Klasse 13), was zum Anstieg des F1 Scores auf 0,85 führt. Am eindrücklichsten ist jedoch der Anstieg innerhalb der Wasserklassen. Durch den Verbund von B423 (Watt), B511 (Wasserlauf), B512 (Wasserfläche) und B521 bis B523 (Lagune, Mündungstrichter und Offenes Meer) konnte ein F1 Score von 0,98 erzielt werden (Abbildung 23). Mit Präzision, Recall und F1 Score nahe 100% ist das Modell in der Lage Wasser gut zu klassifizieren und räumlich auszuweisen. Bei Klasse 14 (Sand, Steine, Erde & Fels) aus Agg16 ist mit F1 von 0,63 ein identischer F1 zur vorherigen Klasse B330 (Sand, Steine, Erde) zu verzeichnen. Sowohl B330 als auch

B332 (Fels) sind wenig vorkommende Klassen mit großen Unterschieden in deren Vorkommen.

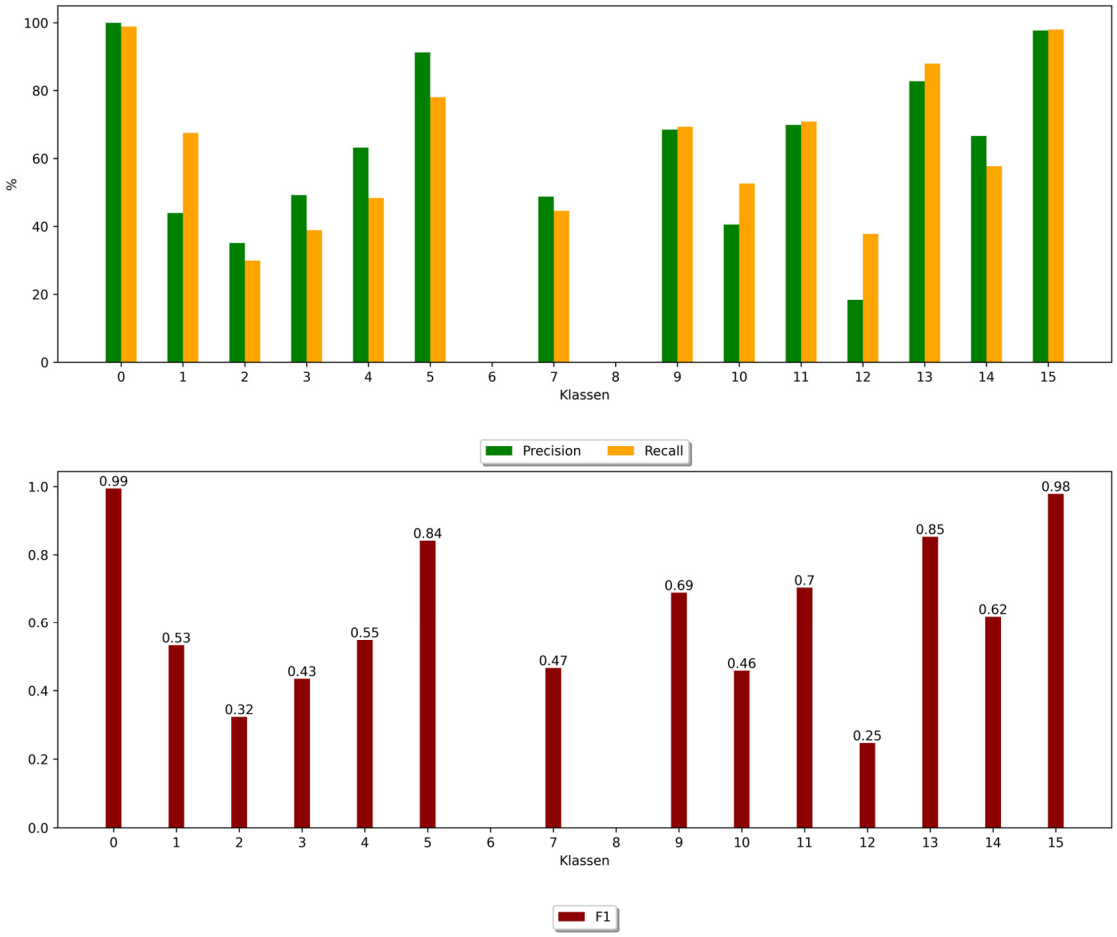


Abbildung 23: Klassenspezifische Gütemetriken für Aggregierungsstufe mit 16 Klassen (Agg16) (2. Szenario). Aufgetragen sind die gemittelten Werte für Präzision und Recall (oben) sowie F1 (unten) für die Kreuzvalidierung über alle Orbits. Dargestellte Klassen sind neben den LBM-Klassen 1 bis 11 verschiedene Aggregierungen der restlichen Klassen. Die Klassen 12 und 13 des LBMs wurden hier in Klasse 12 aggregiert. Weiterhin wurde der Baumbestand (Klasse 14 bis 17 des LBMs) in Klasse 13, die Klassen 18 und 19 (LBM) in Klasse 14 sowie alle Wasserklassen (26 bis 31 des LBMs) in Klasse 15 aggregiert. Die Moorklassen (22 bis 25 des LBMs) wurde zur Klasse 9 hinzugefügt. Neben den NoData Werten beinhaltet Klasse 0 auch die Klasse Brandfläche und Schnee & Eis (vgl. Tabelle 4) (Eigene Darstellung)).

Von 19746km² Landesfläche machen B330 und B332 lediglich 122,91km² (B330) und 0,17km² (B332) aus. Die Metriken werden hier demnach hauptsächlich von B330 bestimmt. Als letzte Veränderung hat die Zusammenführung von homogenem Grünland mit den Klassen der feuchten Flächen keine Veränderung der Resultate für homogenes Grünland gebracht. Homogenes Grünland ist mit 21,36% die zweithäufigste Klasse in SH (Abbildung 14). Aggregiert man nun auf neun Klassen (3. Szenario, siehe Tabelle 8) und weist die Metriken lediglich für die übergeordneten Kategorien aus, erfolgt ein Anstieg des F1 Score für die Kategorie Grünland (Klasse 3, Abb. 24). Der Zusammenschluss von B231 (Homogenes Grünland), B321 (Inhomogenes Grünland) und B233 (Grasland mit

Bäumen <50%) führt zu einem F1 Score von 0,78. Mit knapp 80% ist das Modell in der Lage Grünland als Landbedeckung zu detektieren, trotz der Herausforderungen der Klasse (Kap. 4.1). Die Bebauung (Klasse 1) und feuchte Flächen (Klasse 6) sind mit F1 Scores von 0,46 eher mäßig. Beide Klassen weisen eine höhere Präzision im Vergleich zum Recall auf.

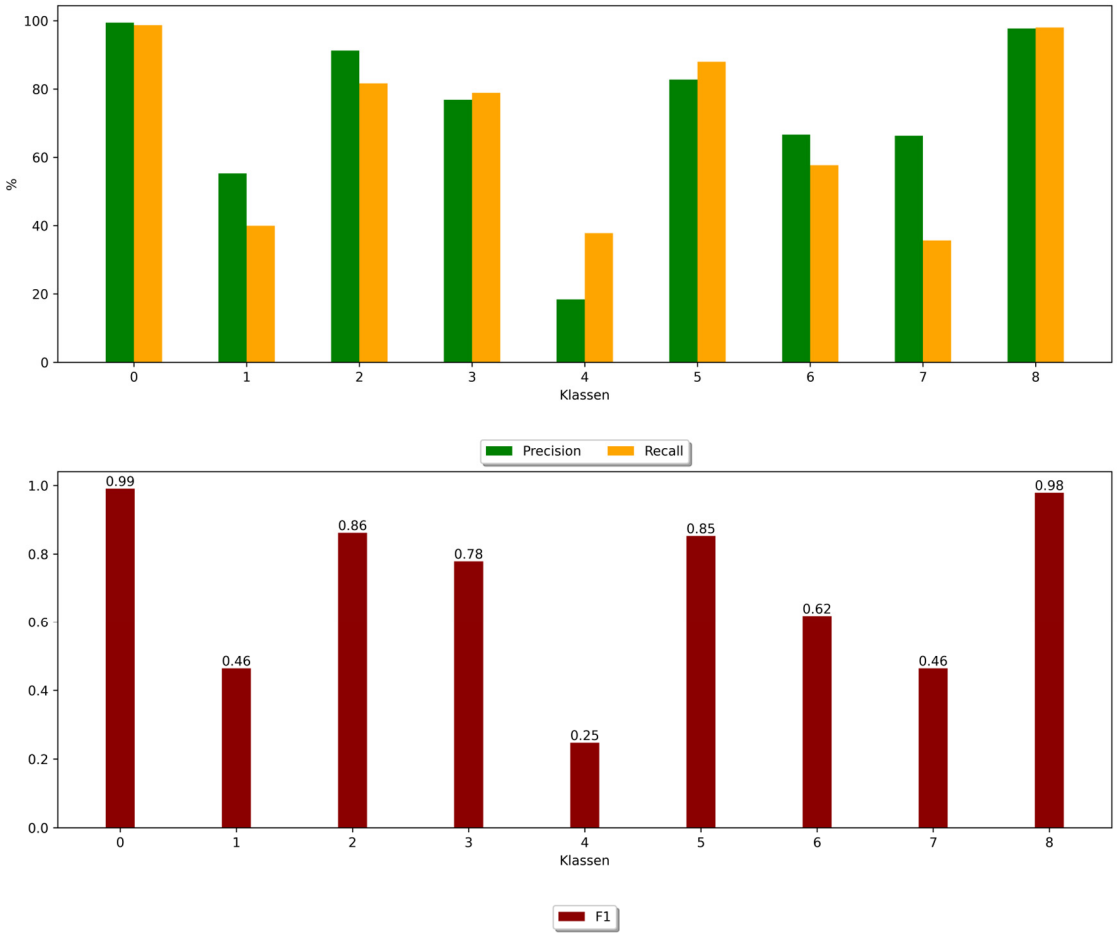


Abbildung 24: Klassenspezifische Gütemetriken für Aggregierungsstufe mit 9 Klassen (Agg9) (3. Szenario). Aufgetragen sind die gemittelten Werte für Präzision und Recall (oben) sowie F1 (unten) für die Kreuzvalidierung über alle Orbits. Dargestellte Klassen sind die LBM-Metaklassen + NoData (NoData (0), Bebauung (1), Ackerland (2), Grünland (3), Sträucher (B322 & B324; 4), Baumbestand (5), Vegetationslose Flächen (6), Feuchte Flächen (7) und Wasserflächen (8) (Eigene Darstellung)).

Nach detailliertem Blick auf die klassenspezifischen Unterschiede soll abschließend noch auf die Gesamtgenauigkeit des Modells bewertet werden. Hierfür wurden Top-K Genauigkeiten für die 32 Klassen Evaluation (1. Szenario) sowie die Genauigkeiten der Aggregierungen (Szenarien 2 und 3) dargestellt. Für das erste Szenario wurden Top-1 bis Top-3 Genauigkeiten angegeben. Bei Top-1 muss die richtige Klasse in der Vorhersage getroffen werden. Bei Top-2 und Top-3 muss die richtige Klasse lediglich unter den zwei

bzw. drei wahrscheinlichsten Klassen zu finden sein, um als korrekt vorhergesagt zu zählen. Hohe Top-K Scores stehen für eine gewisse Robustheit des Modells, da es sich nicht nur auf einzelne Vorhersagen bezieht und Alternativen berücksichtigt. Dies macht das Modell insgesamt zuverlässiger. In Tabelle 9 sind die Gesamtgenauigkeiten für das Modell aufgezeigt. Bei 32 Klassen erreicht das Modell bei Top-1 81,49%, bei Top-2 92,29% und bei Top-3 94,97%. Die Aggregierungsstufen Agg16 und Agg9 erreichen 86,03% und 87,03%. Die Aggregierung von 32 auf 16 Klassen hat demnach eine Steigerung von ca. 4% gebracht, wohingegen zwischen Agg16 und Agg9 lediglich 1% Steigerung erreicht wurde. Zusätzlich wurden für alle Varianten die Genauigkeiten unter Hinzunahme der NoData Klasse präsentiert. Da es bei allen Orbits viele NoData Werte in den Randbereichen der Szenen gibt und diese vom Modell nahezu vollständig erkannt werden, führt die Hinzunahme dieser Klasse zur Steigerung aber gleichzeitig auch Verzerrung der Gesamtgenauigkeit. Bei allen drei Szenarien (32 Klassen, Agg16 und Agg9) ergeben sich Werte von ca. 93-95%.

Des Weiteren zeigt Tabelle 11, dass ein Training von 60 Epochen beim ersten Szenario zur leichten Steigerung der Genauigkeiten kam (Top-1: 81,49% (10) vs. 82,27% (60); Top-2: 92,29% (10) vs. 92,91% (60) und Top-3: 94,97% (10) vs. 95,50% (60)). Bei den anderen beiden Szenarien sind die Ergebnisse nahezu gleich (vgl. Tab. 9).

Tabelle 9: Gesamtgenauigkeiten für die Kreuzvalidierung mit 10 und 60 Epochen (in %). Aufgetragen sind Top-1 bis Top-3 (32 Klassen) sowie die Aggregierungsstufen Agg16 (16 Klassen) und Agg9 (9 Klassen). Zusätzlich ist für Top-1 und für Agg16 und Agg9 einmal die Genauigkeit mit Klasse 0 und ohne Klasse 0 angegeben.

Auswertungsschema	10 Epochen	60 Epochen
Top-1	81,49	82,27
Top-1 (NoData)	93,53	93,83
Top-2	92,29	92,91
Top-3	94,97	95,50
Agg16	86,03	86,04
Agg16 (NoData)	95,10	95,11
Agg9	87,03	87,12
Agg9 (NoData)	95,49	95,53

Um die Gesamtgüte bzw. die klassenspezifischen Genauigkeiten weiter zu verbessern, gibt es mehrere Möglichkeiten. Man könnte mit Augmentierungen arbeiten, um bestehende Daten durch unterschiedliche Verfahren zu vermehren (z.B. Rotationen, Spiegelungen etc.). Da innerhalb des Codes bereits Augmentierungen implementiert sind, wurde sich hier für eine andere Methode entschieden. Durch monatliche Datenbeschaffungskontingente von 5000m² bei Planet Labs, war es möglich weitere Daten zu beziehen. In diesem Zusammenhang wurden drei 5000m² Bereiche in Brandenburg (BB), Hessen (HE) und Mecklenburg-Vorpommern (MV) bezogen. Bei HE und MV erfolgte eine Evaluierung, wie weitere Daten außerhalb von SH die Gesamtgüte des Modells beeinflussen (siehe Anhang 5). Mittels der BB-Daten hingegen, sollte speziell auf unterrepräsentierte Klassen trainiert werden. Hier wurde mittels visuellen Abgleichs mit dem LBM ein Bereich gewählt, der viele Moor/Sumpfflächen enthält, um deren Klassifikationsergebnis dadurch zu verbessern. Die flächenhafte Verteilung der unterschiedlichen Landbedeckungen für BB (Anhang 8) zeigt allerdings, dass die Moorklassen räumlich sehr klein sind und somit wenig Prozent der Gesamtflächen ausmachen. Nichtsdestotrotz wurden sowohl die BB-Daten als auch die aggregierten MVHE-Daten in zwei unterschiedlichen Trainingsläufen mit jeweils 10 Epochen zu den SH-Daten hinzugefügt. Anschließend wurden erneut die klassenspezifischen Metriken ausgegeben. Vergleicht man diese Ergebnisse (Anhang 5) mit den Werten auf der alleinigen Basis der SH-Daten (siehe Abbildung 22) lassen sich kaum Veränderungen feststellen. Es bestehen lediglich kleinere Abweichungen innerhalb einzelner Klassen und keine Klasse zeigte wesentlich bessere oder schlechtere Resultate.

5 Zusammenfassung und Ausblick

Das Ziel dieser Arbeit war neben einem generellen Einstieg in das Potential der künstlichen Intelligenz, die Betrachtung eines praktischen Anwendungsfalls, hier am Beispiel der Ableitung des LBM-DE. Es wurde ein Deep Learning Verfahren genutzt, um die Semantische Segmentierung des Bundeslands Schleswig-Holstein durchzuführen. Als Architektur wurde ein Encoder – Decoder System, mit ResNet50 als Convolutional Backbone und FPN als Decoder, genutzt (Kapitel 2.3). Bei dem eingesetzten Modell handelt es sich um ein vortrainiertes Modell auf der Basis einer deutschlandweiten Abdeckung von RapidEye Daten (Kapitel 3). Da das BKG in Zukunft mit Satellitenszenen verschiedener Sensoren konfrontiert sein wird, sollte in dieser Arbeit evaluiert werden, ob das bestehende DL- Verfahren auch auf Daten anderer Sensoren übertragbar ist. Mit Blick auf die Leitfragen dieser Arbeit werden nun die wichtigsten Erkenntnisse dieser Arbeit zusammengefasst:

1. Welche Gesamt- und klassenspezifischen Genauigkeiten erreicht das KI-Verfahren bei der Klassifizierung von PlanetScope Daten?

Die erreichten Gesamt- und klassenspezifischen Genauigkeiten zeigen das große Potenzial des Verfahrens in der Anwendung auf Erdbeobachtungsdaten. Für eine Landbedeckungsklassifikation mit 32 Klassen erreichte das Verfahren eine Gesamtgenauigkeit von 81,49%. Gerade auch im Hinblick auf die hohe Klassenanzahl ist dieses Ergebnis sehr vielversprechend. Die klassenspezifischen Genauigkeiten sind sehr heterogen. Es gibt sehr gut funktionierende Klassen (gemessen am F1 Score) und weniger gut funktionierende Klassen. Mit einem F1 Score von 0,84 für B211 „Ackerland“ und 0,95 für B523 „Offenes Meer“ sind diese Klassen hier hervorzuheben. Weiterhin gibt es einige Klassen mit F1 Scores zwischen 0,69 und 0,80. Dies umschließt B231 „Grünland“, B233 „Grasland mit Bäumen“, B311 „Laubbäume“, B312 „Nadelbäume“ und B332 „Fels“. Besonders räumlich häufiger vorkommende Klassen mit hohen F1 Scores tragen maßgeblich zur Steigerung der Gesamtgenauigkeit bei (Kapitel 4.2). Neben den erwähnten Klassen weisen einige Klassen niedrigere F1 Scores auf. Dies dient zur Überleitung einiger festgestellter Herausforderungen, mit welchen das Verfahren aufgrund des jetzigen Datenbestands konfrontiert ist. Ein wichtiger Punkt ist die intrakategoriale Unterscheidung der

Landbedeckung. In nahezu allen Landbedeckungskategorien fällt auf, dass die vorliegenden Klassen teils schwierig voneinander zu differenzieren sind. Als Beispiel dienen hier die Unterscheidung von höherer zu niedriger Vegetation, die Mischwaldproblematik, die intrakategoriale Abgrenzung der feuchten oder Wasserklassen sowie die Differenzierung der Bebauung (Kapitel 4.1). Diese Herausforderungen werden im Aktualisierungsprozess des LBMs mit weiteren Daten (z.B. nDOM) oder attributiven Zusätzen zur Landnutzung adressiert, welche dem jetzigen Verfahren nicht vorliegen. Ebenfalls sind manche Klassen (z.B. B313 „Laub- und Nadelbäume“) in der jetzigen Form kaum auflösbar, was mit dem Aktualisierungsprozess des LBMs zusammenhängt (Kapitel 3.3). Einer semantischen Segmentierung auf der alleinigen Basis von Fernerkundungsdaten sind hier Grenzen gesetzt. Neben der Klassenanzahl von 32 Klassen (1.Szenario) wurden zwei Aggregierungsstufen mit 16 (2. Szenario) und 9 Klassen (3. Szenario) evaluiert (Kapitel 4). Speziell die Aggregierung der Grünland- und Wasserklassen, sowie der Klassen der höheren Vegetation sind hier hervorzuheben. Durch die Aggregierung erreichte man hier F1 Scores von 0,78 (Grünland), 0,86 (höhere Vegetation) und 0,98 (Wasser) (vgl. Kapitel 4). Diese hohen Resultate sind ein weiterer Beleg für die Qualität des Verfahrens und den erfolgreichen Domänenwechsel von RapidEye zu PlanetScope.
Neben der Bewertung der Güte wurden in dieser Arbeit die Anpassungen für den Trainingsprozess bei einem Sensorwechsel untersucht (2. Leitfrage). Hierbei spielt Wahl der Hyperparameter eine entscheidende Rolle:

2. Wie müssen die Daten vorverarbeitet werden? Und welche Hyperparameter eignen sich für den Trainingsprozess?

Für das Finetuning mit dem eingesetzten Verfahren ist die richtige Form der Eingangsdaten entscheidend. Da der bestehende Code auf den Einsatz von RapidEye Daten ausgerichtet ist, mussten die Bilddimension der PlanetScope Eingangsdaten angepasst werden. Speziell geht es hier um die Reihenfolge der Anzahl der Spalten und Zeilen, sowie der Bildkanäle. Ähnliche Anpassungen müssen auch für den Übertrag auf weitere Sensoren berücksichtigt werden. Weiterhin mussten die PlanetScope Daten in TOA Reflektanzen umgerechnet und in die verschiedenen Orbits aufgeteilt werden (Kapitel 3.5). Diese wurden anschließend für das Training halbiert, da die genutzte Hardware das komplette Laden der Daten pro Orbit nicht unterstützte. Für die Wahl der Hyperparameter wurde auf

Basis des Testorbits 2457 evaluiert, welche Parameter zu einem guten Kompromiss zwischen Validierungs- und Trainingsverlust führten (Kapitel 3.6). Die Evaluation hat ergeben, dass sich eine Lernrate von 0,001 mit einer Batch Size von 16 und einer Epochenanzahl von 60 als geeignete Hyperparameter erwiesen. Auf Basis dieser Hyperparameter wurde eine Klassifizierung für die Testszene 2457 durchgeführt, die in der höchsten Gesamtgenauigkeit von 84,26% resultierte. Da die Reduktion der Epochenanzahl von 60 auf 10 Epochen lediglich Genauigkeitseinbußen von ca. 1% (siehe Tabelle 9) ergab, wurde das Training dieser Arbeit auf 10 Epochen durchgeführt (Kapitel 3.6).

Abschließend muss noch diskutiert werden, welche Bedeutung die Ergebnisse dieser Arbeit für die Aktualisierung des LBMs haben und wie diese in Zukunft genutzt und verfeinert werden können. Hierzu dient die dritte Leitfrage als Anhaltspunkt:

3. Welche Bedeutung haben die Klassifizierungsergebnisse und wie sind diese Ergebnisse in Bezug auf die Aktualisierung des LBMs zu interpretieren?

Das Verfahren dieser Arbeit bietet entscheidende Vorteile zum jetzigen Aktualisierungsprozess des LBMs. Durch den Einsatz des vortrainierten Modells werden auf Basis der bestehenden LBM-Labels schnell Aussagen über Landbedeckungsveränderungen bei neueren Eingangsdaten getroffen. Weiterhin können durch geringe Anpassungen abweichende Sensortypen verwendet werden. In Zukunft ist davon auszugehen, dass für die Aktualisierung des LBMs die Daten verschiedener Erdbeobachtungsmissionen genutzt werden. In dieser Hinsicht ist der gelungene Sensorwechsel von RapidEye auf PlanetScope sehr vielversprechend. Ein weiterer entscheidender Vorteil ist die Nutzung bestehender Daten ohne händische Generierung von Trainingsgebieten für den Aktualisierungsprozess. Mit dem LBM liegen deutschlandweit bereits hochwertige Labels vor, die für das Training von DL-Verfahren genutzt werden können. Nichtsdestotrotz hat diese Arbeit auch gezeigt, dass der Ableitung der LBM-Klassen auf der alleinigen Basis von Fernerkundungsdaten Grenzen gesetzt sind. Die Klassenbeschreibungen sind oftmals nicht eindeutig, bzw. nicht ohne weitere Schritte mit einer pixelweisen semantischen Segmentierung in der vorliegenden räumlichen Auflösung darstellbar (Kapitel 4.1). Für die Unterscheidung von Klassen mit Vegetationstypen unterschiedlicher Höhe (z.B. B310 und B311/B312) sind detaillierte und aktuelle Höheninformationen essenziell. Hier wäre

die Nutzung eines normalisierten digitalen Oberflächenmodells (nDOM) denkbar. Innerhalb des BKGs liegt eine deutschlandweite Abdeckung mit einer räumlichen Auflösung von 1m/px vor. Denkbar wäre eine Integration dieser Daten als weiterer Bildkanal für die Eingangsdaten. Nach Angleichung der räumlichen Auflösung auf die Auflösung der Eingangsdaten könnten diese zusätzlichen Informationen bei der Differenzierung besagter Klassen innerhalb des Trainingsprozesses helfen.

Für die Unterscheidung von Grünland und den feuchten Flächen (B411 bis B414) wäre der Einsatz von Bodenfeuchteinformationen zu prüfen. Die Evaluation des Klassifizierungsergebnisses hat ergeben, dass die Unterscheidung der beiden Landbedeckungskategorien schwerfällt. Weiterhin ist oftmals zu prüfen, ob ausgewiesene Moor oder Sumpfflächen wirklich noch diesen Klassen angehören. Gerade feuchte Flächen leiden unter den aktuellen klimatischen Veränderungen. Zudem stammen diese Flächen überwiegend aus der Erstableitung aus dem Basis-DLM, bzw. sind nicht Bestandteil der Aktualisierung des LBMs. Denkbar wäre eine generelle Überarbeitung dieser Klassen durch Hinzunahme von Bodenfeuchteinformationen. Hierfür könnten die Daten des Deutschen Wetterdienstes (DWD) verwendet werden, die deutschlandweite Bodenfeuchteinformationen für verschiedene Zeiträume zur Verfügung stellen (DWD, 2024). Im Hinblick auf diese Besonderheiten der Klassen B411 bis B414 sollte den niedrigeren F1 Scores des gesetzten Verfahrens niedrigere Relevanz zugewiesen werden.

Zu den Landbedeckungsklassen des LBM-DE muss generell auf die teilweise schwierig zu modellierende Klassenbeschreibung hingewiesen werden. Als ein Beispiel dient hier die Klasse Bebauung, die vom Verfahren in begrünten Vorstadtbereichen häufig der Klasse B233 „Grasland mit Bäumen“ zugeordnet wird (Kapitel 4.1). Diese Vorhersage scheint im ersten Moment auf ein sehr ungenaues Modell hinzuweisen, was bei Sichtung der beiden Klassen nicht der Realität entspricht. Beide Klassen sind in grünen Vorstadtbereichen spektral recht ähnlich. Schaut man sich die Ergebnisse in den Stadtkernen an, wo der Anteil an grün abnimmt, sagt das Modell die richtige Klasse vorher.

Solche Klassen würden durch eine höhere Auflösung, sowie räumlich differenzierterer Labels profitieren, sodass Vegetation von urbanen Strukturen abgegrenzt werden kann.

Eine solche Steigerung der räumlichen Auflösung könnte durch Nutzung weiterer Fernerkundungsdaten realisiert werden. Hier ist speziell die geplante Nutzung von Vision-1 Daten mit einer panchromatischen Auflösung von 87cm/px hervorzuheben (ESA 2024). Als Alternative könnte die Nutzung von Orthophotos mit 20cm/px in Betracht gezogen

werden. Die gesteigerte räumliche Auflösung könnte weiterhin zur besseren Erfassung von Struktur und Aussehen der unterschiedlichen Landbedeckungen führen. Am Beispiel von Aufforstungsflächen (B310) unterscheiden sich diese von ausgewachsenen Bäumen (B311/B312) in Form und Struktur. Hier müsste evaluiert werden, ob die Nutzung dieser Daten zu besseren Ergebnissen führt und, ob diese die zusätzliche Rechenlast rechtfertigen. Mit der aktuellen Hardwareumgebung innerhalb des BKGs wäre eine solche Prozessierung für die gesamte Bundesfläche eher unrealistisch.

Zum Schluss soll noch ein wesentlicher Aspekt zur Steigerung der Modellgüte erwähnt werden. Hierbei handelt es sich um eine Postprozessierung der Klassifizierungsergebnisse des Verfahrens. Als mögliches Beispiel dient hier die Klasse B313 „Laub- und Nadelbäume“. Die Aktualisierung des LBMs geschieht auf Basis bestehender Polygone (Kapitel 3.3). Bei B313 darf keine Baumart mehr als 75% der Fläche einnehmen und es müssen mindestens 50% mit Bäumen bestanden sein. Durch Hinzunahme der bestehenden Polygongrenzen könnte das Klassifizierungsergebnis dieser Arbeit verwendet werden, um diese prozentualen Anteile in der Postprozessierung zu berechnen. Dies würde wahrscheinlich zur besseren Ausweisung der Klasse B313 führen. Dieses Beispiel soll für die vielfältige Nutzbarkeit des Klassifizierungsergebnisses sensibilisieren. Die niedrigen F1 Scores bestimmter Klassen sprechen nicht für einen unzureichenden Ansatz. Eher sprechen sie für die Diskrepanz zwischen Klassenbeschreibung und deren Aktualisierung und den Möglichkeiten des unverarbeiteten Klassifizierungsergebnisses des DL-Ansatzes dieser Arbeit. In Zukunft muss am BKG evaluiert werden, wie die Klassifizierungsergebnisse weiterverarbeitet und genutzt werden, und wie sich durch den geänderten methodischen Ansatz auch die Klassenzusammensetzung des LBMs verändert. Weiterhin ist die Hinzunahme angesprochener Zusatzdaten zu testen. Der methodische Ansatz dieser Arbeit hat gezeigt, dass durch geringe Anpassungen verschiedene Fernerkundungsdaten mit hoher Gesamtgenauigkeit (81,49%) genutzt werden können. Ohne große manuelle Anpassungen können teils sehr genaue Landbedeckungsveränderungen ermittelt werden. Speziell im Hinblick auf die Nutzung verschiedener Sensortypen aus unterschiedlichen Erdbeobachtungsmissionen sind diese Ergebnisse sehr vielversprechend.

Literaturverzeichnis

1. ABADI, M., BARHAM, P., CHEN, J., CHEN, Z., DAVIS, A., DEAN, J., ... & ZHENG, X. (2016). {TensorFlow}: a system for {Large-Scale} machine learning. In *12th USENIX symposium on operating systems design and implementation (OSDI 16)* (pp. 265-283).,

2. ALBUMENTATIONS 2024: ALBUMENTATIONS: FAST AND FLEXIBLE IMAGE AUGMENTATIONS | URL https://albumentations.ai/docs/ (Letzter Zugriff: 11.01.2024)

3. ANACONDA (2024): ANACONDA INC | URL https://www.anaconda.com/download (Letzter Zugriff: 10.01.2024)

4. BADRINARAYANAN, V., KENDALL, A., & CIPOLLA, R. (2017). Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE transactions on pattern analysis and machine intelligence*, *39*(12), 2481-2495.

5. BKG (2023): DOKUMENTATION DES LBM-DE 2018 | URL https://sg.geodaten-zentrum .de/web_public/gdz/dokumentation/deu/lbm-de2018.pdf (Letzter Aufruf: 12.12.2023)

6. BOGUSZEWSKI, A., BATORSKI, D., ZIEMBA-JANKOWSKA, N., DZIEDZIC, T., & ZAMBRZYCKA, A. (2021). LandCover. ai: Dataset for automatic mapping of buildings, woodlands, water and roads from aerial imagery. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (pp. 1102-1110).

7. BREIMAN, L. (2001). Random forests. *Machine learning*, *45*, 5-32.

8. CAFFE (2024): Deep Learning Framework der University of California | URL https://caffe. berkeleyvision.org/ (Letzter Zugriff: 08.12.2023)

9. CAI, B.; JIANG, Z.; ZHANG, H.; YAO, Y.; NIE, S. (2018): Online Exemplar-Based Fully Convolutional Network for Aircraft Detection in Remote Sensing Images. IEEE Geosci. Remote Sens. Lett. **2018**, 15, 1095–1099.

10. CAMPS-VALLS, G., X. X. ZHU, D. TUIA UND M. REICHSTEIN (2021). „Introduction“. In: *Deep Learning for the Earth Sciences. A Comprehensive Approach to Remote Sensing, Climate Science and Geosciences*. Hrsg. von G. Camps-Valls, X. X. Zhu, D.Tuia und M. Reichstein. Wiley, S. 1–11. isbn: 978-1-1196-4614-3.

11. CHEN, L. C., PAPANDREOU, G., SCHROFF, F., & ADAM, H. (2017). Rethinking atrous convolution for semantic image segmentation. arXiv preprint arXiv:1706.05587.

12. CIRESAN, D.; MEIER, U.; SCHMIDHUBER, J. (2012): Multi-column deep neural networks for image classification. In Proceedings of the 2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Providence, RI, USA, 16–21 June 2012; pp. 3642–3649.

13. CNTK (2024): MICROSOFT COGNITIVE TOOLKIT (DEEP LEARNING FRAMEWORK | URL https://github.com/microsoft/CNTK (Letzter Zugriff: 09.01.2024)

14. DAHL, G.E.; YU, D.; DENG, L.; ACERO, (2012): A. Context-Dependent Pre-Trained Deep Neural Networks for Large-Vocabulary Speech Recognition. Trans. Audio Speech and Lang. Proc. **2012**, 20, 30–42.

15. DALAL, N., & TRIGGS, B. (2005, JUNE). Histograms of oriented gradients for human detection. In *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)* (Vol. 1, pp. 886-893). Ieee.

16. DAUDT, R.C.; LE SAUX, B.; BOULCH, A.; GOUSSEAU, Y. (2018): Urban Change Detection for Multispectral Earth Observation Using Convolutional Neural Networks. In Proceedings of the IGARSS 2018—2018 IEEE International Geoscience and Remote Sensing Symposium, Valencia, Spain, 22–27 July 2018; pp. 2115–2118.

17. DAUPHIN, Y. N., FAN, A., AULI, M., & GRANGIER, D. (2017, JULY). Language modeling with gated convolutional networks. In *International conference on machine learning* (pp. 933-941). PMLR.

18. DEMIR, I., KOPERSKI, K., LINDENBAUM, D., PANG, G., HUANG, J., BASU, S., ... & RASKAR, R. (2018). Deepglobe 2018: A challenge to parse the earth through satellite images. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops (pp. 172-181).

19. DENG, J., DONG, W., SOCHER, R., LI, L. J., LI, K., & FEI-FEI, L. (2009, JUNE). Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition* (pp. 248-255). Ieee.

20. DERU, M., & NDIAYE, A. (2019). *Deep Learning mit TensorFlow, Keras und TensorFlow. js*. Rheinwerk Verlag.

21. DLR (2023): Beschreibung der RapidEye Mission | URL https://www.dlr.de/rd/desktopdefault .aspx/tabid-2440/3586_read-5336 (Letzter Aufruf: 10.12.2023)

22. DWD (2024): Bodenfeuchte Deutschland | URL https://www.dwd.de/DE/leistungen/bofeu_analyse/bfana. html;jsessionid=78EEC8B926F76AB4B 6BB3D7094C35689 .live31093?nn =16102 (Letzter Zugriff: 04.02.2024)

23. ESA (2024): Vision-1 Beschreibung | URL https://earth.esa.int/eogateway/catalog/vision-1-full-archive-and-tasking (Letzter Zugriff: 06.02.2024)

24. FORTSCHREIBUNG DER NATIONALEN KI- STRATEGIE (2020): FORTSCHREIBUNG DER NATIONALEN KI-STRATEGIE DER BUNDESREGIERUNG DEUTSCHLAND | URL. https://www.ki-strategie-deutschland.de/home.html?file=files/downloads /201201_Fortschreibung_KI-Strategie.pdf&cid=947 (Letzter Zugriff: 12.06.2023)

25. GARBIN, C., ZHU, X., & MARQUES, O. (2020). Dropout vs. batch normalization: an empirical study of their impact to deep learning. *Multimedia Tools and Applications*, *79*, 12777-12815.

26. GAßNER, K. (2019): Maschinelles Lernen für die IT-Sicherheit. In V. Wittpahl, editor, *Künstliche Intelligenz*, pages 72–91. Springer Berlin Heidelberg, Berlin, Heidelberg, 2019.

27. GDAL (2024): GEOSPATIAL DATA ABSTRACTION LIBRARY FÜR PYTHON | URL https://gdal.org/api/python_bindings.html#building-as-part-of-the-gdal-library-source-tree (Letzter Zugriff: 08.01.2024)

28. GOODFELLOW, I., BENGIO, Y., & COURVILLE, A. (2016). *Deep learning*. MIT press.

29. GORDON-RODRIGUEZ, E., LOAIZA-GANEM, G., PLEISS, G., & CUNNINGHAM, J. P. (2020). Uses and abuses of the cross-entropy loss: Case studies in modern deep learning.

30. HAO, S., ZHOU, Y., & GUO, Y. (2020). A brief survey on semantic segmentation with deep learning. *Neurocomputing*, *406*, 302-321.

31. HE, K., ZHANG, X., REN, S., & SUN, J. (2015). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 770-778).

32. HE, K., ZHANG, X., REN, S., & SUN, J. (2016). Identity mappings in deep residual networks. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part IV 14* (pp. 630-645). Springer International Publishing.

33. HENRY, C.J.; STORIE, C.D.; PALANIAPPAN, M.; ALHASSAN, V.; SWAMY, M.; ALESHINLOYE, D.; CURTIS, A.; KIM, D. (2019): Automated LULC map production using deep neural networks. Int. J. Remote Sens. **2019**, 40, 4416–4440.

34. HOESER, T., & KUENZER, C. (2020): Object detection and image segmentation with deep learning on earth observation data: A review-part i: Evolution and recent trends. *Remote Sensing*, *12*(10), 1667.

35. HOESER, T., BACHOFER, F., & KUENZER, C. (2020). Object detection and image segmentation with deep learning on Earth observation data: A review—Part II: Applications. *Remote Sensing*, *12*(18), 3053.

36. HUANG, L., JIANG, B., LV, S., LIU, Y., & FU, Y. (2020). Deep Learning-based Semantic Segmentation of Remote Sensing Images: A Survey. IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing.

37. IOFFE, S., & SZEGEDY, C. (2015, JUNE). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning* (pp. 448-456). pmlr.

38. JÉGOU, S., DROZDZAL, M., VAZQUEZ, D., ROMERO, A., & BENGIO, Y. (2017). The one hundred layers tiramisu: Fully convolutional densenets for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops* (pp. 11-19).

39. KAPLAN, J. (2017): *Künstliche Intelligenz*. MITP, Frechen, 1. Auflage Edition, 2017.

40. KEMKER, R., SALVAGGIO, C., & KANAN, C. (2018). Algorithms for semantic segmentation of multispectral remote sensing imagery using deep learning. ISPRS journal of photogrammetry and remote sensing, 145, 60-77.

41. KERAS (2024): Deep Learning for humans. High-Level API für Tensorflow | URL https://keras.io/ (Letzter Zugriff: 15.01.2024)

42. KINGMA, D. P., & BA, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

43. KIRSTE, M. AND M. SCHÜRHOLZ (2019). Entwicklungswege zur KI. In V. Wittpahl, editor, *Künstliche Intelligenz*, pages 21–35. Springer Berlin Heidelberg, Berlin, Heidelberg, 2019.

44. LATEEF, F., & RUICHEK, Y. (2019). Survey on semantic segmentation using deep learning techniques. Neurocomputing, 338, 321-348.

45. LECUN Y., L. BOTTOU, Y. BENGIO, AND P. HAFFNER (1999). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1999.

46. LIGHTNING (2024): HIGH-LEVEL API FÜR PYTORCH | URL https://lightning.ai/docs/pytorch/stable/starter/introduction.html (Letzter Zugriff: 15.01.2024)

47. LIN, G., MILAN, A., SHEN, C., & REID, I. (2017). Refinenet: Multi-path refinement networks for high-resolution semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 1925-1934).

48. LONG, J., SHELHAMER, E., & DARRELL, T. (2015). Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 3431-3440).

49. LOWE, D. G. (2004). Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, *60*, 91-110.

50. MA, L., LIU, Y., ZHANG, X., YE, Y., YIN, G., & JOHNSON, B. A. (2019). Deep learning in remote sensing applications: A meta-analysis and review. ISPRS journal of photogrammetry and remote sensing, 152, 166-177.

51. MARMANIS, D., DATCU, M., ESCH, T., & STILLA, U. (2016). Deep learning earth observation classification using ImageNet pretrained networks. IEEE Geoscience and Remote Sensing Letters, 13(1), 105-109.

52. MATHEW, A., AMUDHA, P., & SIVAKUMARI, S. (2021). Deep learning techniques: an overview. *Advanced Machine Learning Technologies and Applications: Proceedings of AMLTA 2020*, 599-608.

53. MINAR, M. R., & NAHER, J. (2018). Recent advances in deep learning: An overview. *arXiv preprint arXiv:1807.08169*.

54. MNIH, V. (2013). Machine learning for aerial image labeling. University of Toronto (Canada).

55. MONETT, D., LEWIS, C. W., & THÓRISSON, K. R. (2020). Introduction to the JAGI Special Issue 'On Defining Artificial Intelligence'–Commentaries and Author's Response. *Journal of Artificial General Intelligence*, *11*(2), 1-100.

56. MÜLLER, R., KORNBLITH, S., & HINTON, G. E. (2019). When does label smoothing help?. *Advances in neural information processing systems*, *32*.

57. NAIR, V., & HINTON, G. E. (2010). Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)* (pp. 807-814).

58. NATIONALE KI STRATEGIE (2018): NATIONALE KI-STRATEGIE DER BUNDESREGIERUNG DEUTSCHLAND | URL https://www.ki-strategie-deutschland.de/home.html?file =files/downloads/Nationale_KI-Strategie.pdf&cid=728 (Letzter Zugriff: 12.06.2023)

59. NUMPY (2024): NUMERICAL PYTHON. SCIENTIFIC COMPUTATION LIBRARY | URL https://numpy.org/doc/stable/user/whatisnumpy.html (Letzter Zugriff: 29.12.2023)

60. NWANKPA C., W. IJOMAH, A. GACHAGAN, AND S. MARSHALL (2018). Activation Functions: Comparison of trends in Practice and Research for Deep Learning, 08.11.2018.

61. OGUNDOKUN, R. O., MISRA, S., DOUGLAS, M., DAMAŠEVIČIUS, R., & MASKELIŪNAS, R. (2022). Medical internet-of-things based breast cancer diagnosis using hyperparameter-optimized neural networks. *Future Internet*, *14*(5), 153.

62. PINHEIRO, P. O., LIN, T. Y., COLLOBERT, R., & DOLLÁR, P. (2016). Learning to refine object segments. In Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part I 14 (pp. 75-91). Springer International Publishing.

63. PLANET LABS (2023(A)): DOKUMENTATION ZUR RAPIDEYE DATA PROZESSIERUNG | https://developers.planet.com/docs/data/rapideye/#processing (Letzter Zugriff: 10.12.2023)

64. PLANET LABS (2023(B)): PRODUKT BESCHREIBUNG ZUR PLANETSCOPE SATELLITENMISSION | URL https://assets.planet.com/docs/Planet_Combined_Imagery_Product_Specs_letter_screen.pdf (Letzter Aufruf: 11.12.2023)

65. PLANET LABS (2023(C)): SENSOR ÜBERSICHT PLANETSCOPE | URL https://developers .planet.com/docs/data/planetscope/#constellation-and-sensor-overview (Letzter Aufruf: 12.12.2023)

66. POWERS, D. M. (2020). Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation. *arXiv preprint arXiv:2010.16061*.

67. PRIYANKA, N, S., LAL, S., NALINI, J., REDDY, C. S., & DELL'ACQUA, F. (2022). DIResUNet: Architecture for multiclass semantic segmentation of high resolution remote sensing imagery data. Applied Intelligence, 52(13), 15462-15482.

68. PYCHARM (2024): PYTHON-IDE VON JETBRAINS | URL https://www.jetbrains.com/de-de/pycharm/ (Letzter Zugriff: 17.01.2024)

69. PYTHON (2024): PROGRAMMIERSPRACHE DER PYTHON FOUNDATION | URL https://www.python.org/ (Letzter Zugriff 17.01.2024)

70. PYTORCH (2024): DEEP LEARNING FRAMEWORK | URL https://pytorch.org/docs/stable/index.html (Letzter Zugriff: 17.01.2024)

71. RONNEBERGER, O., FISCHER, P., & BROX, T. (2015). U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18* (pp. 234-241). Springer International Publishing.

72. ROSCHER UND DREES (2022): Einführung in die künstliche Intelligenz und das maschinelle Lernen. In: GRUNAU, W. (ED.). (2022). *Künstliche Intelligenz in Geodäsie und Geoinformatik: Potenziale und Best-Practice-Beispiele*. Wichmann, H.

73. ROTTENSTEINER, F., SOHN, G., JUNG, J., GERKE, M., BAILLARD, C., BENITEZ, S., & BREITKOPF, U. (2012). The ISPRS benchmark on urban object classification and 3D building reconstruction. ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences; I-3, 1(1), 293-298.

74. RUDER, S. (2017). An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*.

75. S. RUSSELL AND P. NORVIG (2015). *Artificial Intelligence: A Modern Approach*. Pearson Education India, 3 edition, 2015. ISBN 978-93-325-4351-5

76. SAFE (2024): FME FORM VON SAFE SOFTWARE | URL https://fme.safe.com/platform /#fme-form (Letzter Zugriff: 17.01.2024)

77. SAMMUT, C., & WEBB, G. I. (EDS.). (2011). *Encyclopedia of machine learning*. Springer Science & Business Media.

78. SCHÜRHOLZ M. AND E.-C. SPITZNER (2019). Hardware für KI. In V. Wittpahl, editor, *Künstliche Intelligenz*, pages 36–47. Springer Berlin Heidelberg, Berlin, Heidelberg, 2019.

79. SCIKIT-LEARN (2024): MACHINE LEARNING IN PYTHON | URL https://scikit-learn.org/stable/ (LETZTER ZUGRIFF: 17.01.2024)

80. SEFERBEKOV, S., IGLOVIKOV, V., BUSLAEV, A., & SHVETS, A. (2018). Feature pyramid network for multi-class land segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops* (pp. 272-275)

81. SEGMENTATION MODELS (2024): PYTHON LIBRARY WITH NEURAL NETWORKS FOR IMAGE SEGMENTATION | URL https://segmentation-models.readthedocs.io/en/latest/ tutorial.html (Letzter Zugriff 17.01.2024)

82. SEMMELMANN, K. (2021). Was ist Clustering? Definition, Methoden und Beispiele | Data Driven Company, 2021. URL https://datadrivencompany.de/was-ist-clustering-definition-methoden-und-beispiele/. Letzter Zugriff 25.11.2023.

83. SHAO, Z., YANG, K., & ZHOU, W. (2018). Performance evaluation of single-label and multi-label remote sensing image retrieval using a dense labeling dataset. Remote Sensing, 10(6), 964.

84. SHARMA, A., LIU, X., YANG, X., & SHI, D. (2017). A patch-based convolutional neural network for remote sensing image classification. Neural Networks, 95, 19-28.

85. SHARMA, S. & ATHAIYA, A. (2017). Activation functions in neural networks. *Towards Data Sci*, *6*(12), 310-316.

86. SHERRAH, J. (2016). Fully convolutional networks for dense semantic labelling of high-resolution aerial imagery. arXiv preprint arXiv:1606.02585.

87. SHORTEN, C., & KHOSHGOFTAAR, T. M. (2019). A survey on image data augmentation for deep learning. *Journal of big data*, *6*(1), 1-48.

88. SILVER, DAVID; SCHRITTWIESER, JULIAN; SIMONYAN, KAREN; ANTONOGLOU, IOANNIS; HUANG, AJA; GUEZ, ARTHUR ET AL. (2017): Mastering the game of Go without human knowledge. In: Nature 550 (7676), S. 354–359. DOI: 10.1038/nature24270.

89. SPACENET. SPACENET 2: BUILDING DETECTION V2. Available online: https://github.com/SpaceNetChallenge/BuildingDetectors_Round2 (Letzter Zugriff: 17.01.2024).

90. TANG, T.; ZHOU, S.; DENG, Z.; LEI, L.; ZOU, H. (2017): Arbitrary-Oriented Vehicle Detection in Aerial Imagery with Single Convolutional Neural Networks. Remote Sens. **2017**, 9, 1170.

91. TAYLOR, J. R., & THOMPSON, W. (1982). *An introduction to error analysis: the study of uncertainties in physical measurements* (Vol. 2, pp. 193-200). Mill Valley, CA: University science books.

92. TENSORFLOW (2024): Deep Learning Framework von Google | URL https://www.tensorflow.org/ learn (Letzter Zugriff 17.01.2024)

93. TERVEN, J., CORDOVA-ESPARZA, D. M., RAMIREZ-PEDRAZA, A., & CHAVEZ-URBIOLA, E. A. (2023). Loss functions and metrics in deep learning. A review. *arXiv preprint arXiv:2307.02694*.

94. THEANO (2024): MACHINE LEARNING LIBRARY FÜR PYTHON. WAS IST THEANO? | URL https://www.bigdata-insider.de/was-ist-theano-a-705862/#:~:text=Theano%20ist%20eine%20Library%20f%C3%BCr,das%20Machine%20Learning%20eingesetzt%20werden. (LETZTER ZUGRIFF: 15.01.2024)

95. VAN BEERS, F., LINDSTRÖM, A., OKAFOR, E., & WIERING, M. (2019, MARCH). Deep neural networks with intersection over union loss for binary image segmentation. In *Proceedings of the 8th International Conference on Pattern Recognition Applications and Methods* (pp. 438-445). SciTePress.

96. VERLEYSEN, M., & FRANÇOIS, D. (2005, JUNE). The curse of dimensionality in data mining and time series prediction. In *International work-conference on artificial neural networks* (pp. 758-770). Berlin, Heidelberg: Springer Berlin Heidelberg.

97. VOELSEN, M., M. TEIMOURI, F. ROTTENSTEINER UND C. HEIPKE (2022). „Investigating 2D and 3D Convolutions for Multitemporal Land Cover Classification using Remote Sensing Images“. In: *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences* 3, S. 271–279. doi: 10.5194/isprs-annals-V-3-2022-271-2022.

98. WANG, C., & ZHONG, C. (2021). Adaptive feature pyramid networks for object detection. *IEEE Access*, *9*, 107024-107032.

99. WANG, H., WANG, Y., ZHANG, Q., XIANG, S., & PAN, C. (2017). Gated convolutional neural network for semantic segmentation in high-resolution images. Remote Sensing, 9(5), 446.

100. WANG, Z., & XUE, X. (2014). Multi-class support vector machine. *Support vector machines applications*, 23-48.

101. WEI, Y.;WANG, Z.; XU, M. (2017): Road Structure Refined CNN for Road Extraction in Aerial Image. IEEE Geosci. Remote Sens. Lett. **2017**, 14, 709–713.

102. WURM, M., DROIN, A., STARK, T., GEIß, C., SULZER, W., TAUBENBÖCK, H. (2021): *Deep Learning-based generation of building stock data from remote sensing for urban heat demand modeling*. ISPRS International Journal of Geo-Information, vol. 10(1) 23, https://doi.org/10.3390/ijgi10010023

103. WURM, M., T. STARK, X. X. ZHU, M. WEIGAND UND H. TAUBENBÖCK (2019). „Semantic segmentation of slums in satellite images using transfer learning on fully convolutional neural networks“. In: *ISPRS Journal of Photogrammetry and Remote Sensing* 150, S. 59–69. doi: 10.1016/j.isprsjprs.2019.02.006.

104. WUTTKE. L (2020). Machine Learning: Definition, Algorithmen, Methoden und Beispiele. *Datasolut GmbH*, 11.08.2020. URL https://datasolut.com/was-ist-machine-learning. (Letzter Zugriff 22.12.2023)

105. YANG, H.L.; YUAN, J.; LUNGA, D.; LAVERDIERE, M.; ROSE, A.; BHADURI, B. (2018): Building Extraction at Scale Using Convolutional Neural Network: Mapping of the United States. IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens. **2018**, 11, 2600–2614.

106. YOU, Y.; LI, Z.; RAN, B.; CAO, J.; LV, S.; LIU, F. (2019): Broad Area Target Search System for Ship Detection via Deep Convolutional Neural Network. Remote Sens. **2019**, 11, 1965.

107. YU, H., YANG, Z., TAN, L., WANG, Y., SUN, W., SUN, M., & TANG, Y. (2018). Methods and datasets on semantic segmentation: A review. *Neurocomputing*, *304*, 82-103.

108. ZHAI, Y., ROY, D. P., MARTINS, V. S., ZHANG, H. K., YAN, L., & LI, Z. (2022). Conterminous United States Landsat-8 top of atmosphere and surface reflectance tasseled cap transformation coefficients. *Remote Sensing of Environment*, *274*, 112992.

109. ZHANG, A., LIPTON, Z. C., LI, M., & SMOLA, A. J. (2023). *Dive into deep learning*. Cambridge University Press.

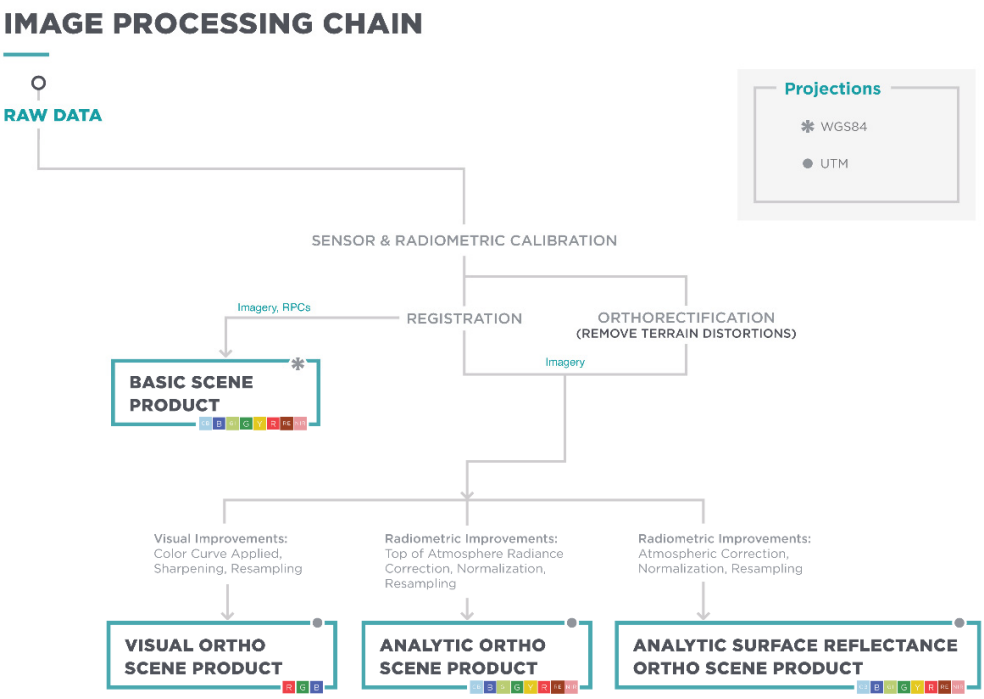
110. ZHANG, C.; SARGENT, I.; PAN, X.; LI, H.; GARDINER, A.; HARE, J.; ATKINSON, P.M. (2019): Joint Deep Learning for land cover and land use classification. Remote Sens. Environ. **2019**, 221, 173–187.

111. ZHANG, G., LEI, T., CUI, Y., & JIANG, P. (2019). A dual-path and lightweight convolutional neural network for high-resolution aerial image segmentation. ISPRS International Journal of Geo-Information, 8(12), 582.

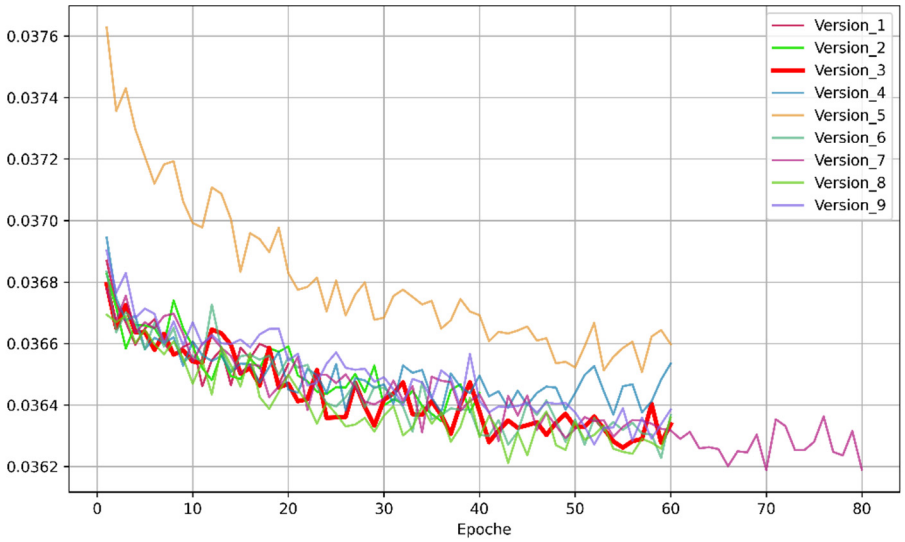
112. ZHANG, M., HU, X., ZHAO, L., LV, Y., LUO, M., & PANG, S. (2017). Learning dual multi-scale manifold ranking for semantic segmentation of high-resolution images. Remote Sensing, 9(5), 500.

113. ZHU, H., MENG, F., CAI, J., & LU, S. (2016). Beyond pixels: A comprehensive survey from bottom-up to semantic image segmentation and cosegmentation. *Journal of Visual Communication and Image Representation*, *34*, 12-27.

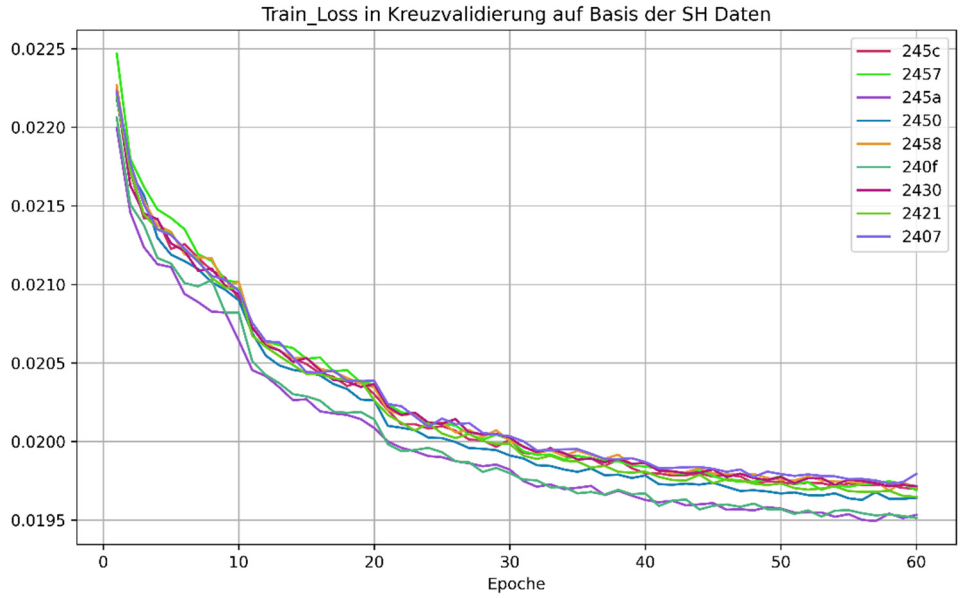
Anhang



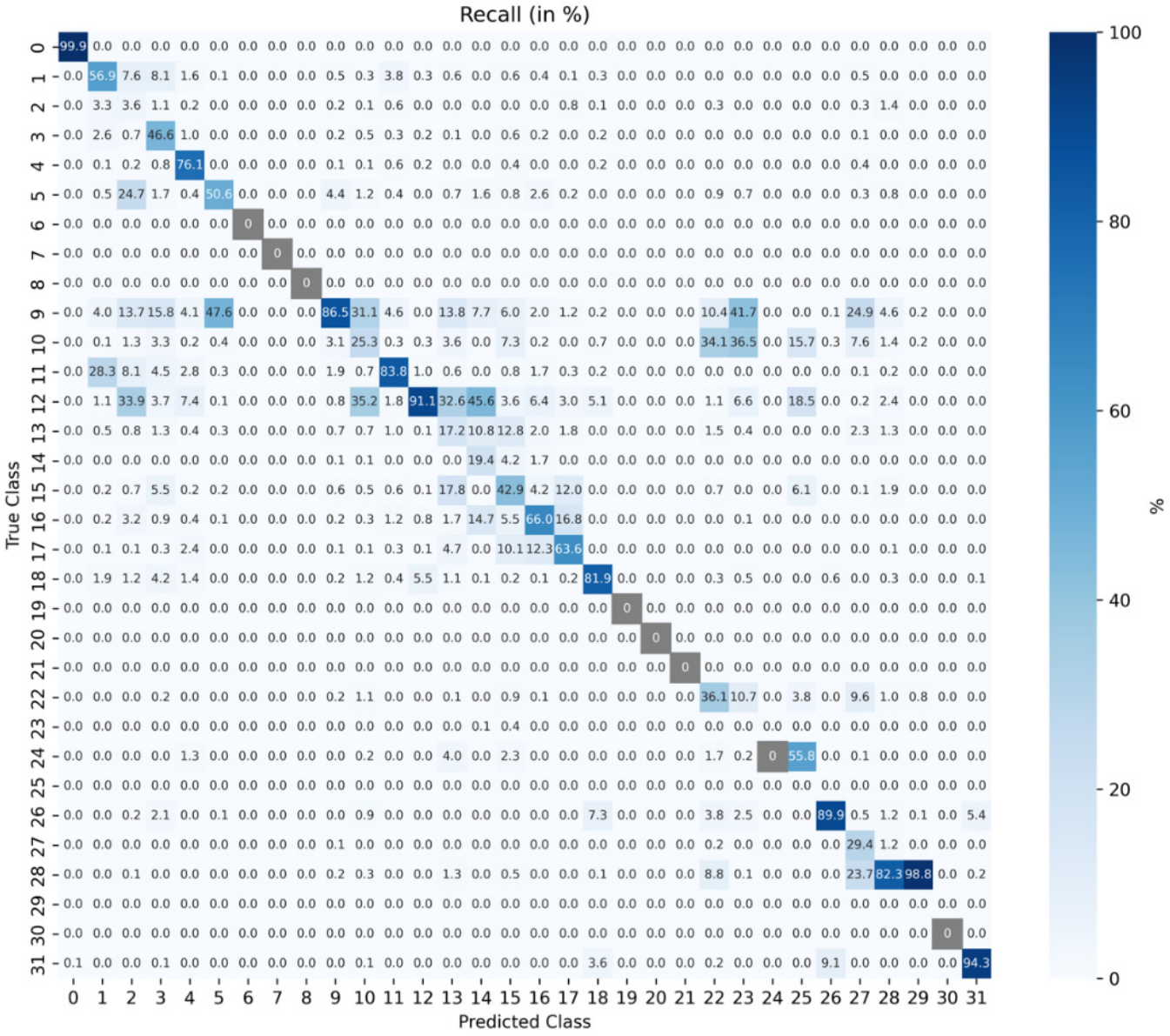
Anhang 1:Überischt der Prozessierungsstufen der PlanetScope Daten. Unterschieden wird zwischen dem Basic Scene Product (lediglich Sensor und Radiometrische Kalibrierung) und den orthorektifizierten Produkten (Visual Ortho Scene Product, Analytic Ortho Scene Product und Analytic Surface Reflectance Ortho Scene Product) mit unterschiedlichen radiometrischen oder visuellen Verbesserungen (PLANET LABS (2023(C)).



Anhang 2: Trainingsverlust auf Testorbit 2457 während Hyperparameteroptimierung. Aufgetragen sind unterschiedlichen Testläufe (Version_X; siehe Tabelle 6) und deren Entwicklung des Trainingsverlusts während des Trainings. Hyperparameteroptimierung wurde mit exkludierter Testszene bei der Gewichtsberechnung durchgeführt. Die Versionen sind farblich differenziert, wobei Version 3 hervorgehoben wird (fett, rot), aufgrund deren geeigneter Hyperparameter für das spätere Training (Eigene Darstellung).

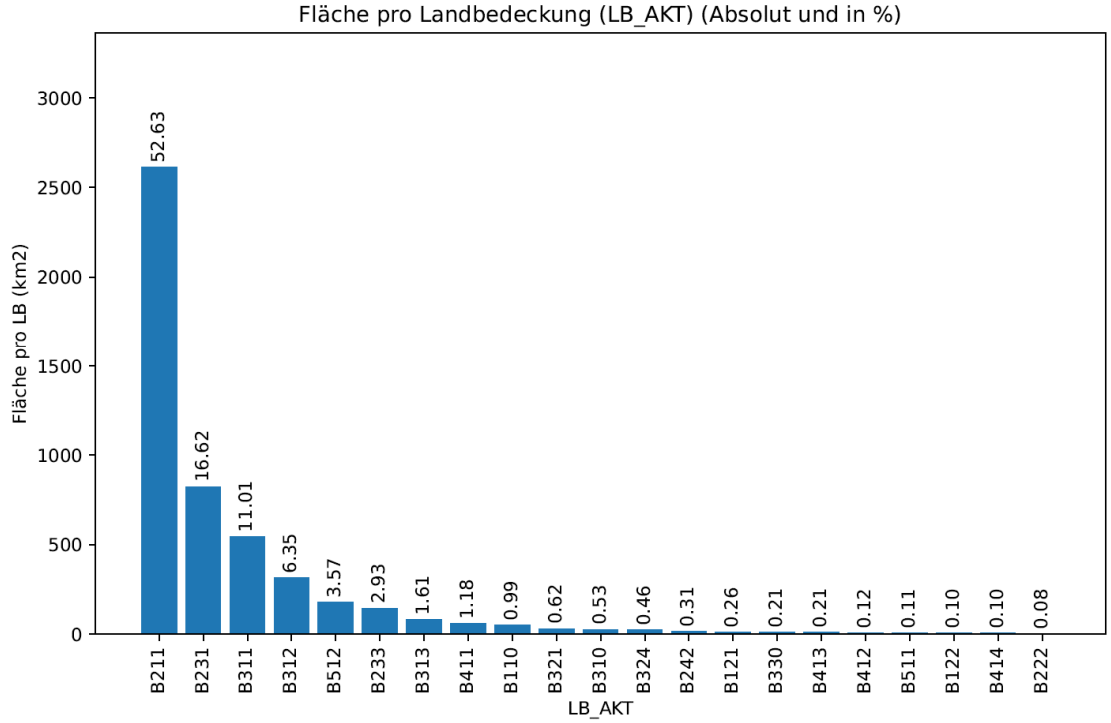


Anhang 3: Trainingsverlust aller Testorbits innerhalb der Kreuzvalidierung auf 60 Epochen. Farbliche Zuweisung der Orbit-ID. Für die Berechnung der Bildstatistiken und der Gewichte wurde die Testszene inkludiert (Eigene Darstellung).

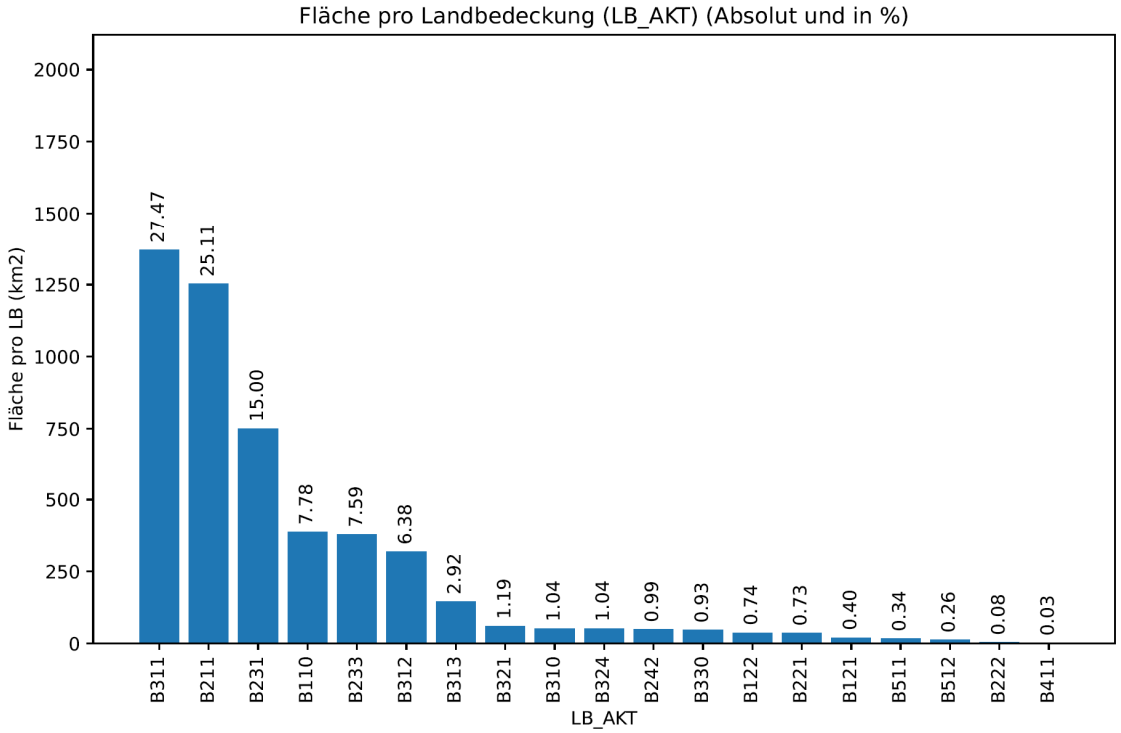


Anhang 4: Konfusionsmatrix für den Recall auf Basis des Testorbits 245c. Aufgetragen sind wahren Klassen (True Class) und die vorhergesagten Klassen (Predicted Class) für die 31 Landbedeckungsklassen des LBM-DE (und NoData als Klasse 0). Der Recall ist in Prozent dargestellt, wobei graue Felder auf nicht vorhandene Klassen (oder NoData) hinweisen (Eigene Darstellung).

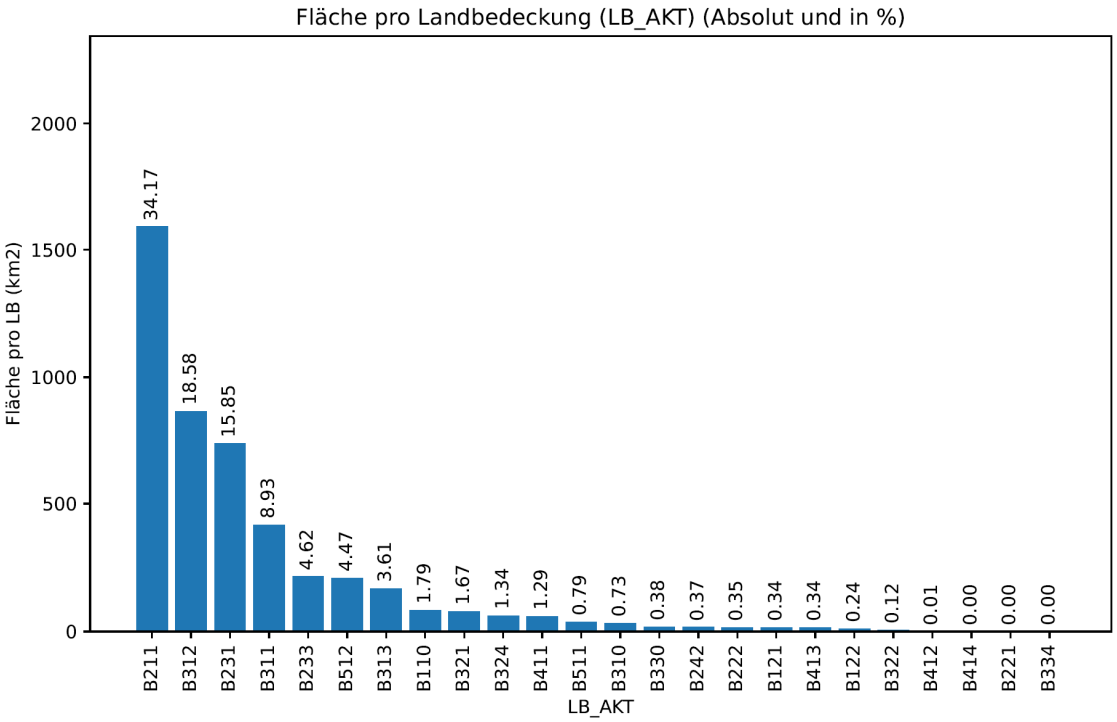
Anhang 5: Gemittelte klassenspezifische Gütemetriken pro Landbedeckungsklasse des LBM-DE auf Basis der SH-Daten sowie ausgewählter Gebiete in HE, MV und BB (Details zur Klassenverteilung der Zusatzdaten sind Anhang 6, 7, und 8 zu entnehmen). Aufgetragen sind die gemittelten Werte für Präzision und Recall (1. Plot) und F1 (2. Plot) für SH + BB sowie Präzision und Recall (3. Plot) und F1 (4. Plot) für SH + MVHE. Mittels dieser zusätzlichen Trainingsdaten wurde erneut eine Kreuzvalidierung über alle Orbits von SH durchgeführt. Die Klassenzuweisung zwischen Modell (M) und LBM ist der Tabelle zu entnehmen. Die Klassen 6, 8, 20, und 21 sind im Untersuchungsgebiet nicht vorhanden (Eigene Darstellung).



Anhang 6: Fläche pro Landbedeckung der Zusatzdaten in Mecklenburg-Vorpommern (MV). Aufgetragen sind die Landbedeckungsklassen des LBM-DE. Zusätzlich zur absoluten Fläche wurde deren prozentualer Anteil an der Gesamtfläche hinzugefügt (Eigene Darstellung).



Anhang 7: Fläche pro Landbedeckung der Zusatzdaten in Hessen (HE). Aufgetragen sind die Landbedeckungsklassen des LBM-DE. Zusätzlich zur absoluten Fläche wurde deren prozentualer Anteil an der Gesamtfläche hinzugefügt (Eigene Darstellung).



Anhang 8: Fläche pro Landbedeckung der Zusatzdaten in Brandenburg (BB). Aufgetragen sind die Landbedeckungsklassen des LBM-DE. Zusätzlich zur absoluten Fläche wurde deren prozentualer Anteil an der Gesamtfläche hinzugefügt (Eigene Darstellung).

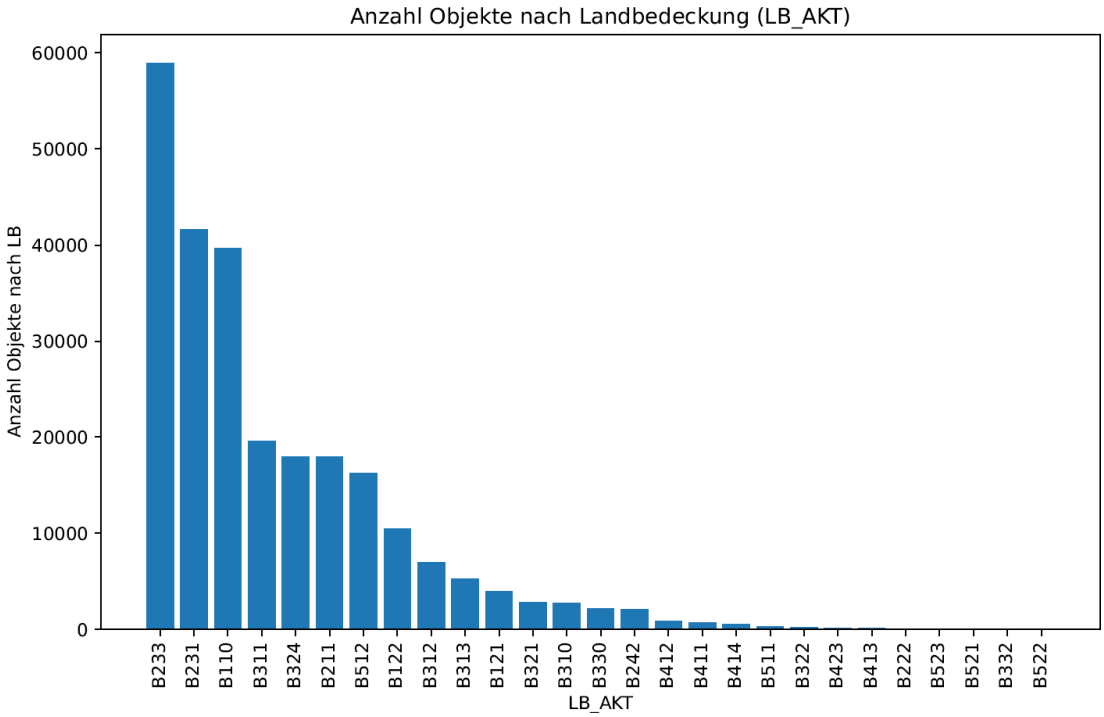
Bezeichnung Trainingsgebiete	Ausweisung der Trainingsgebiete (TA) mittels LB und LN Klassen, sowie weiterer Attribute und Indizes)
lockere Bebauung	TA 112 : (B 233 ||B 231 ||B 242) & (N 142 ||N 112) & SIE > 30
Hallen, Gebäude:	TA 110
Nadelwald	TA 312 : (VEG ≥ 95 & (N 999 ||N 311))
Totholz	TA 312 t: (VEG ≥ 95 , HKT 2 ,NDVI)
Laubwald	TA 311 : (VEG ≥ 95 & (N 999 ||N 311))
Bäume < 5 m	TA 310 : (B 310 ||B 324)
Wasser	TA 511 : (B 511 ||B 512 ||B 523) & (N 999 ||N 510)
Acker (nackter Boden)	TA 211
Acker (mit Vegetation)	TA 211 t
Weinberge, Hopfen, Hopfen,……:	TA 224 (B 221 ||B 222 ||B 224) (nur mit Vegetation)
Nackter Boden	TA 330 : (B 330 ||B 332) kein N 133 & kein ZUS = W “
Baustelle	TA 133
Grünland:	TA 231

Anhang 9: Beschreibung der Trainingsgebiete, die im jetzigen Aktualisierungsprozess des LBM-DE generiert werden. Neben der Bezeichnung der Trainingsgebiete sind verwendete Landbedeckungs- (B) und Landnutzungsklassen (N) aufgelistet. Zusätzlich spielen bei manchen Trainingsgebieten noch Vegetations- (VEG) und Versiegelungsgrade (SIE), sowie weitere Indizes eine Rolle (Eigene Darstellung).

Code (LBM)	Code (M)	Beschreibung
-	0	NoData
B110	1	Flächen mit Bebauungsstruktur und Verkehrsnetz.
B121	2	Flächen, auf denen sich Anlagen oder spezielle Bauwerke zur Erzeugung, bzw. Verteilung von Elektrizität, Wärme und Wasser oder Kläranlagen befinden.
B122	3	Flächen, deren Oberfläche mit Asphalt, Beton, Pflaster etc. versiegelt ist.
B242	4	Anthropogen geprägte Mischflächen, die eine regelmäßige Struktur besitzen. Es muss ein Wechsel zwischen mind. 3 verschiedenen Bedeckungsklassen vorliegen, von den eine versiegelt sein muss. Schreber-/Kleingartenanlagen, (Freizeit- und Vergnügungsparks, Zoos – sofern sie der Beschreibung entsprechen), Campingplätze und Friedhöfe
B211	5	Regelmäßig (mind. 1x pro Jahr) gepflügte, meist im Fruchtwechsel bewirtschaftete Flächen. Flächen zum Anbau von Getreide, Gemüse, Futterpflanzen, Industriepflanzen und Hackfrüchten.
B221	6	Mit Weinreben bestockte Flächen
B222	7	Parzellen mit Obstbäumen und -sträuchern, auf denen eine oder mehrere Obstsorten angebaut werden. Plantagenstruktur
B224	8	Felder mit Gerüsten, auf denen Hopfen angebaut wird
B231	9	Grünlandflächen mit durchgehendem Grasbestand, die regelmäßig beweidet oder gemäht werden
B321	10	Grünlandflächen, die höchstens einmal jährlich bearbeitet werden. Unregelmäßiges Erscheinungsbild, häufig mit Stauden und Gestrüpp durchsetzt.
B233	11	Grünlandflächen, die zu maximal 50% von Bäumen bestanden sind.
B322	12	Heidelandschaft mit Zwergsträuchern, d.h. mit Büschen, Sträuchern, Kräutern und / oder geringwertigem Baumbestand (< 50%). Es existiert eine niedrige und geschlossene Vegetationsdecke. Der Boden ist trocken und sandig. Oft handelt es sich um (ehemalige) Truppenübungsplätze.
B324	13	Busch- oder Strauchvegetation mit einzelnen Bäumen. Die Flächen können entweder aus Waldflächen durch allmähliche Degenerierung oder durch natürliche Verjüngung des Waldes entstanden sein (vereinzelte junge Bäume bis zu einer Baumhöhe von 5m sind möglich).
B310	14	Waldflächen, die mit Bäumen bis zu einer Höhe von 5m bestanden sind (entweder aufgeforstet oder durch Naturverjüngung entstanden).
B311	15	Flächen, die zu mindestens 50% mit Bäumen bestanden sind. Von diesen Bäumen müssen mindestens 75% Laubbäume sein.
B312	16	Flächen, die zu mindestens 50% mit Bäumen bestanden sind. Von diesen Bäumen müssen mindestens 75% Nadelbäume sein.
B313	17	Flächen, die zu mindestens 50% mit Bäumen bestanden sind. Keine Waldart darf mehr als 75% dieser Fläche ausmachen. Es muss eine baumweise oder baumgruppenweise Durchmischung von Laub- und Nadelbäumen erkennbar sein.
B330	18	Nicht versiegelte, vegetationsarme Flächen mit lockerem Gestein, Erde und/oder Sand als Oberfläche. Sie können anthropogen geprägt sein wie Deponie- und Lagerflächen, Industriebrachen und Baustellen oder auch natürlich wie trockene, steppenartige Flächen, alpine Tundra und Erosionsflächen.
B332	19	Felsen und anstehendes Gestein.
B334	20	Flächen, auf denen es kürzlich gebrannt hat und die zum größten Teil noch Brandspuren aufweisen
B335	21	Von Gletschern und Dauerschnee bedeckte Flächen.

B411	22	Unbewaldete Flächen, die teilweise, vorübergehend oder ständig feucht sind. Ursache hierfür kann fließendes oder stehendes Wasser sein. Tief liegende Flächen, die im Winter normalerweise überflutet und ganzjährig mit Wasser gesättigt sind
B412	23	Unbewaldete, nasse oder feuchte Flächen, deren Boden vorwiegend aus Torfmoos und unvollständig abgebauten pflanzlichen Stoffen besteht. Torfmoore können abbaubar oder nicht abbaubar sein.
B413	24	Sümpfe gemäß Definition von B411 mit Büschen und/oder Bäumen (Anteil 30% - 50% à der Bewuchs ist zusätzlich zum Sumpf zu verstehen; die Definition besagt nicht, dass bei einem Anteil von 30% Bewuchs nur 70% Sumpf vorliegen. Vielmehr ist davon auszugehen, dass der Sumpf sich unter den Büschen/Bäumen fortsetzt. Dennoch erfolgt eine Erfassung als B31x bei einem Anteil von > 50%).
B414	25	Moore gemäß Definition von B412 mit Büschen und/oder Bäumen (Anteil 30% - 50% à der Bewuchs ist zusätzlich zum Moor zu verstehen; die Definition besagt nicht, dass bei einem Anteil von 30% Bewuchs nur 70% Sumpf vorliegen. Vielmehr ist davon auszugehen, dass das Moor sich unter den Büschen/Bäumen fortsetzt. Dennoch erfolgt eine Erfassung als B31x bei einem Anteil von > 50%). Keine aktiven, aber stillgelegte oder in Renaturierung befindliche Abbauflächen.
B423	26	Flächen im Küstenbereich mit Schlamm, Sand und Felsen, die sich zwischen den Niveaus des mittleren Hoch- und mittleren Niedrigwasserstands befinden und somit bei Ebbe trockenfallen, in der Regel ohne Vegetation.
B511	27	Natürliche oder künstlich angelegte Gewässerläufe, die dem Wasserabfluss dienen. Dazu gehören auch Kanäle.
B512	28	Stehende, natürliche oder künstliche Gewässer (Süßwasser). Nicht zugehörig sind Überschwemmungsflächen, es gelten nur dauerhafte Wasserflächen. Auch Seitenarme von Flüssen mit stehendem Gewässer.
B521	29	Salz- oder Brackwasserzonen im Küstenbereich, die vom Meer durch eine Landzunge o.ä. getrennt sind. Diese Wasserflächen können an wenigen Stellen eine Verbindung mit dem Meer haben. Die Verbindung kann ständig oder nur periodisch (z.B. Jahreszeitenbedingt oder in Abhängigkeit der Tide) bestehen. Eine Lagune ist meist geprägt von geringer Wassertiefe (ca. 3 - 8 m). Dazu zählen auch die Bodden an der Ostseeküste
B522	30	Verbreiterter Teil einer Flussmündung (Trichtermündung) ins Meer, der dem Einfluss der Gezeiten ausgesetzt ist.
B523	31	Der an den mittleren Niedrigwasserstand angrenzende Bereich des offenen Meeres

Anhang 10: Beschreibung der LBM-DE Klassen. Neben den IDs für die einzelnen Landbedeckungsklassen vom BKG (BXXX) und Modell (0-31) sind Beschreibungen der Flächen, die der jeweiligen Landbedeckungsklasse zugeordnet werden, aufgelistet.



Anhang 11: Anzahl der Objekte pro Landbedeckungsklasse im Untersuchungsgebiet Schleswig-Holstein aus dem LBM-DE (Eigene Darstellung).

Eidesstattliche Erklärung

Hiermit erkläre ich, dass ich die vorliegende Masterarbeit

„Das Potenzial von künstlicher Intelligenz am Beispiel der Ableitung des Landbedeckungsmodells (LBM-DE) für das Bundesland Schleswig-Holstein“

selbständig ohne fremde Hilfe angefertigt habe. Ich habe nur die in der Arbeit ausdrücklich benannten Quellen und Hilfsmittel benutzt. Wörtlich oder sinngemäß übernommenes Gedankengut habe ich als solches kenntlich gemacht.

Ort, Datum	Unterschrift