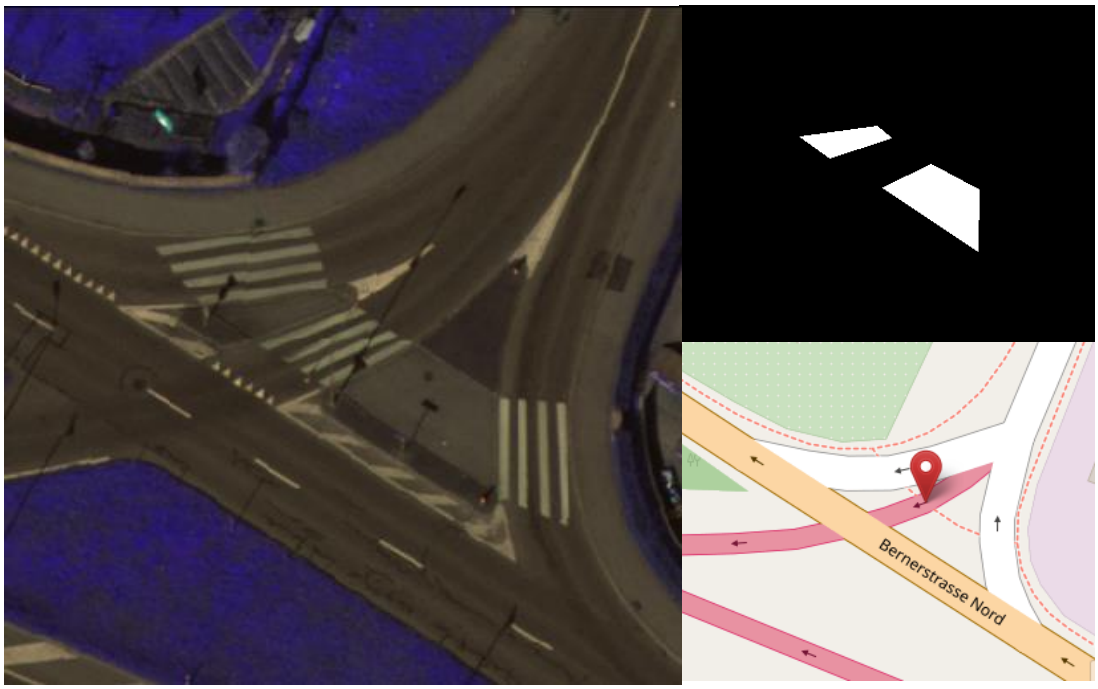
Segmentierung von Fussgänger-Inseln in hochaufgelösten Luftbildern und anschliessendem OpenStreetMap Abgleich



Project Thesis 1
Master of Science in Engineering
Profile Data Science
OST Campus Rapperswil
Spring Semester 2022

Author: Lucien Hagmann
Advisor: Stefan Keller

Rapperswil 20.07.2022

Quellennachweis Titelblatt:

- OSM Screenshot (https://www.openstreetmap.org – vom 15.07.2022)
- MS Engineering Logo (https://www.msengineering.ch/ - vom 30.03.2022)
- OST Logo (https://www.ost.ch/de/ - vom 30.03.2022)

Abstract

Due to advances in the machine learning domain, it is now possible to process large amounts of data in an efficient and precise manner. Consequently, there is great potential to apply these technologies in areas where a lot of data is still processed and checked manually. Satellite images and other aerial images for geographic information systems are nowadays produced at a rate that makes it almost impossible to check them manually. However, these tools rely on actuality. A computerized approach might be able to help with this challenge.

In this paper, such an approach is documented. Based on computer vision theory, an already existing machine learning model -an artificial neural network- is extended and trained for the detection of refuge islands in aerial images. Furthermore, the coordinates of these objects are extracted and prepared for comparison with the geographic database of OpenStreetMap.

The project contains three main phases. In phase one the dataset is cleaned and prepared to make sure the later used artificial neural network is fed with useable data. This step also includes the creation of labels that are needed by the model to check whether it fulfills the given task correctly. Phase two contains the whole training process and testing process of the model. At the end of this step, the model will have produced predictions for a subset of the data on whether there are refuge islands located on these images. In phase three, the coordinates of these predictions are stored in a geoJSON file. Subsequently, the locations are compared with already mapped pedestrian islands on OpenStreetMap all over Switzerland.

The result of this work is a computer-aided method for object segmentation in aerial photographs. This can then be used to support the verification of geodata. Due to the correct preparation of the orthophotos, as well as a suitable configuration and targeted training of the artificial neural network, the model used, is now able to reliably recognize pedestrian islands. The subsequent comparison showed that all predicted objects by the model are already correctly mapped in OpenStreetMap.

The dataset comprises of over 4000 orthophotos provided by the Swiss Federal Office of Topography. For the practical part of this project, the programming language Python, and the deep learning framework of Fastai are used. The code and additional comments are stored in Juypter Notebook files. The repository is available under: https://github.com/LucienOST/PT1-Refuge_Islands

Keywords: Computer Vision, Semantic Segmentation, Convolutional Neural Networks, Geograpic Information Systems, OpenStreetMap, Pedestrian Islands

Management Summary

Durch die Fortschritte der letzten Jahre im Bereich des Maschinellen-Lernens ist es heute möglich, enorme Datenmengen automatisiert und mit hoher Genauigkeit zu analysieren. Entsprechen gross ist das Potential, diese Technologie dort einzusetzen, wo aufwendige Überprüfungssaufgaben noch manuell durchgeführt werden. Ein wichtiger Teilbereich dieser Disziplin bildet Computer Vision, welcher sich insbesondere mit Daten in Form von Bildern und Videos befasst. Für Geoinformationssysteme werden heute Satellitenbilder und Luftaufnahmen aus Flugzeugen oder Drohnen in einem Umfang erstellt, welche es praktisch unmöglich macht, alle diese Daten manuell zu überprüfen. Diese Systeme sind jedoch auf Akutalität angewiesen. Ein computergestütztes Verfahren kann hier abhelfen.

Ausgangslage

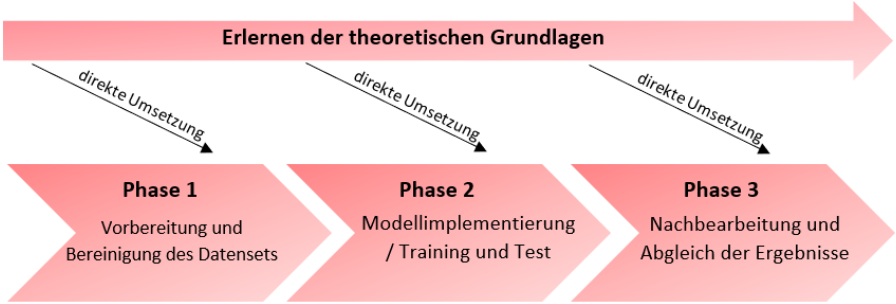
Diese Arbeit befasst sich mit einem Implementierungsansatz für eine solche Aufgabe. Gestützt auf die Theorie des Maschinellen-Lernens, soll ein bereits existierendes Modell (ein künstliches neuronales Netz) weiterentwickelt werden. Das Ziel ist es, in hochaufgelösten Luftbildern (Orthofotos) Mittelinseln von Fussgängerstreifen zu erkennen und deren Position mittels geographischer Koordinaten abzuspeichern. In der Nachbearbeitung sollen die extrahierten Informationen mit Daten aus OpenStreetMap abgeglichen und auf Unterschiede überprüft werden.

Ziele

1. Aneignung der Grundkonzepte des Maschinellen-Lernens im Bereichen Computer Vision, mit Fokus auf Bildsegmentierung mittels Convolutional Neural Networks.
2. Aneignung nützlicher Grundlagen im Bereich Geoinformationssysteme und Geodaten.
3. Anpassung des bestehenden Modells sowie anschliessendes Training mit Bildern von Fussgängerinseln.
4. Testen des Modells und Auswertung der Ergebnisse.
5. Anhand der erzielten Ergebnisse einen Ansatz zum Abgleich mit OSM-Daten erstellen.
6. Dokumentation des gesamten Implementierungsprozesses und des Datenflusses. Vom Erhalt des Datensatzes bis zum Abgleich mit OSM.

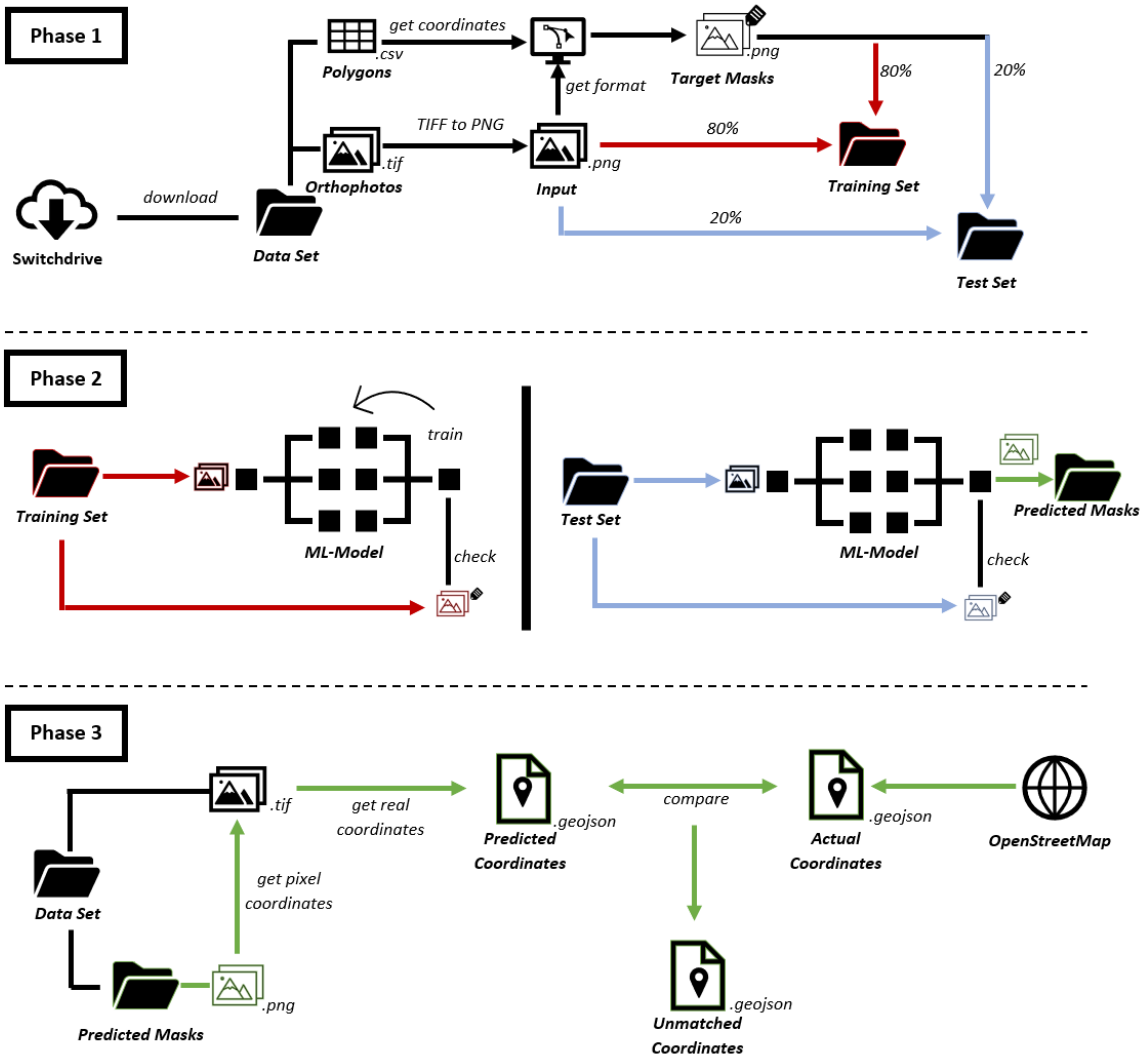
Vorgehensweise

Das Projekt wird in drei Hauptphasen unterteilt. Zu Beginn jeder Phase werden zuerst die notwendigen Grundkonzepte mittels einer Literaturrecherche und Online-Tutorials erlernt. Anschliessend wird dieses Wissen direkt im Praxisteil angewandt. Für die Umsetzung wird die Programmiersprache Python und das Deep Learning Framework Fastai verwendet.



Ergebnisse

Das Ergebnis dieser Arbeit ist ein computergestütztes Verfahren zur Objektsegmentierung in Luftbildern. Dieser kann anschliessend zur Unterstützung bei der Überprüfung von Geodaten verwendet werden. Durch die korrekte Aufbereitung der Orthofotos, sowie durch eine passende Konfiguration und gezieltem Training des künstlichen neuronalen Netzes, ist das verwendete Modell nun fähig, Fussgängerinseln zuverlässig zu erkennen. Zudem konnten weitere Information aus den Luftbildern extrahiert werden. Der gesamte Datenfluss über die drei Projektphasen lässt sich anhand von folgendem Diagramm ableiten:



Die implementierten Prozessschritte der Nachbearbeitung ermöglichen es, diese Information direkt für eine erste Analyse weiterzuverwenden. Ein erster Abgleich mit OSM für verschiedene Regionen der Schweiz wurde durchgeführt. Für die Vorhersagen des neuronalen Netzes konnte festgestellt werden, dass alle erkannten Mittelinseln für Fussgängerstreifen - welche sich in der Schweiz befinden- in OSM bereits korrekt erfasst sind.

Inhaltsverzeichnis

1 Einleitung

Diese Projektarbeit findet im Rahmen des Master of Science in Engineering (MSE) an der Ostschweizer Fachhochschule (OST) statt. Das vom Autor gewählte MSE-Profil ist Data Science. Entsprechend dieser Vertiefung soll durch diese Arbeit ein praxisnahes und anwendungsbezogenes Forschungsthema bearbeitet werden.

Data Science umfasst ein breites Spektrum an wissenschaftlich fundierten Methoden, Prozessen, Algorithmen und Systemen, um aus Daten Informationen zu gewinnen. Ein wichtiger Teilbereich dieses Fachgebietes bildet Computer Vision, welcher sich insbesondere mit Daten in Form von Bildern und Videos befasst (Matt Przybyla 2020). Hauptziel dieses Fachbereiches ist es, Objekte oder Muster zu erkennen, zu segmentieren und einer Klasse zuzuordnen. Durch die Fortschritte der letzten Jahre im Bereich des Maschinellen-Lernens (ML) ist es heute möglich, enorme Datenmengen automatisiert und mit hoher Genauigkeit zu analysieren. Auch Punkto Geschwindigkeit übertreffen die Fähigkeiten solcher Algorithmen die eines Menschen bereits um ein vielfaches. Entsprechen gross ist das Potential, diese Technologie dort einzusetzen, wo aufwendige Überprüfungssaufgaben noch manuell durchgeführt werden (Ilija Mihajlovic 2019).

Ein vielversprechendes Anwendungsgebiet für Computer Vision Algorithmen ist der Bereich Geoinformationssysteme (GIS). Satellitenbilder und Luftaufnahmen von Flugzeugen und Drohnen werden heute in einem Umfang erstellt, welche es praktisch unmöglich macht, alle diese Daten auf herkömmliche Art und Weise zu analysieren. Ein computergestützter Ansatz kann hier abhelfen (Rohit Singh 2019).

1.1 Problemstellung

Geodatenbanken sind auf Aktualität und Richtigkeit ihrer Daten angewiesen. Nur so kann die Realität akkurat in einem Kartographie-Tool abgebildet werden. Das Projekt OpenStreetMap (OSM) stellt solche Kartendaten frei zur Verfügung. OSM setzt hierbei auf das kollektive und lokale Wissen seiner Nutzerinnen und Nutzer, welche selbständig Beiträge erfassen und pflegen. Zur Verifizierung der Korrektheit dieser Daten, nutzen Autoren unter anderem Luftbilder, GPS-Geräte und Feldkarten (OSM Org. 2022). Ein automatisierter Standard, welcher die OSM-Gemeinschaft bei diesem Verifizierungsprozess unterstützt existiert noch nicht.

Diese Arbeit befasst sich mit einem Implementierungsansatz für eine solche Aufgabe. Gestützt auf die Theorie des Maschinellen-Lernens, soll ein bereits existierendes ML-Modell weiterentwickelt werden. Das Ziel: In hochaufgelösten Luftbildern Fussgängerinseln zu erkennen und deren Position mittels geographischer Koordinaten abzuspeichern. In einem weiteren Schritt sollen die extrahierten Geoinformation mit den Daten von OSM abgeglichen und auf Unterschiede überprüft werden.

Beim erwähnten Modell handelt es sich um ein künstliches Neuronales Netz, welches mit Luftbildern weitertrainiert werden soll. Diese Luftbilder, in dieser Arbeit auch Orthofotos genannt, wurden im Vorfeld dieser Arbeit beim Bundesamt für Landestopographie (Swisstopo) beschafft.

1.1.1 Vorarbeit

Diese Arbeit ist eine Fortsetzung einer abgeschlossenen Projektarbeit von Kevin Ammann, welcher ebenfalls an der OST ein Masterstudium absolviert. Von Kevin Ammann wurde bereits ein computergestützter Prozess erarbeitet, um ebenfalls Informationen aus Luftbildern zu extrahieren. Dieses Verfahren bildet die Basis des hier vorliegenden Projektes, entsprechend oft wird auf diese verwiesen. War das Ziel des ersten Implementierungsansatzes noch die Detektion von Fussgängerstreifen, wird in dieser Arbeit nun der Fokus auf Fussgängerinseln gelegt. Durch die sehr guten Ergebnisse der Vorarbeit, kann direkt an diese angeknüpft werden. Der Programmiercode und das verwendete ML-Modell können teilweise wiederverwendet werden. Um viele Wiederholungen zu vermeiden, liegt der Fokus in diesem Bericht speziell auf jenen Bereichen, welche sich von der Vorarbeit unterscheiden, oder in dieser nur gering oder gar nicht behandelt wurden. Ein Hauptziel dieser Arbeit ist es, den Bericht von Kevin Ammann zu ergänzen und zu vervollständigen. Gewisse Überschneidungen sind jedoch nicht zu verhindern. S.A. Literaturverzeichnis: (Kevin Ammann 2022)

1.2 Betreuung und Vorbereitung

Betreut wird dieses Projekt vom Institut für Software an der OST, genauer, dem Geometa Lab unter der Leitung von Prof. Stefan Keller. Als Experten für moderne Softwaretechnik in Verbindung mit geowissenschaftlichen Anwendungen unterstützen diese den Autor dieser Arbeit. Ausserdem geht die Aufgabenstellung auf das Geometa Lab zurück, welches selbst eine Vielzahl an Projekten im Bereich Open Data (OpenStreetMap) unterhält.

1.3 Zielsetzung

Das erarbeitete Verfahren von Kevin Ammann soll für das Erkennen von Fussgänger Inseln ausgeweitet werden. Jedoch unterscheiden sich die Datensätze, welche für das Trainieren des Modells verwendet werden, inhaltlich und auch in ihrer Struktur. Daher müssen einige Prozessschritte, überarbeitet oder sogar neu hinzugefügt werden. Da diese noch nicht in der Vorarbeit enthalten sind, muss hierfür auch das notwendige theoretische Wissen angeeignet und dokumentiert werden. Ebenfalls neu, soll in der Nachbearbeitung ein erster Abgleich mit OpenStreetMap erfolgen. Zusammengefasst ergeben sich folgende Ziele für diese Projektarbeit:

1. Aneignung der Grundkonzepte des Maschinellen-Lernens im Bereichen Computer Vision, mit Fokus auf Bildsegmentierung mittels Convolutional Neural Networks.
2. Aneignung nützlicher Grundlagen im Bereich Geoinformationssysteme und Geodaten.
3. Anpassung des bestehenden Modells sowie anschliessendes Training mit Bildern von Fussgängerinseln.
4. Testen des Modells und Auswertung der Ergebnisse.
5. Anhand der erzielten Ergebnisse einen Ansatz zum Abgleich mit OSM-Daten erstellen.
6. Dokumentation des gesamten Implementierungsprozesses und des Datenflusses. Vom Erhalt des Datensatzes bis zum Abgleich mit OSM.

1.4 Aufbau und Vorgehen

Das Aneignen einer theoretischen Basis ist notwendig, damit die jeweiligen Schritte im Praxisteil dieser Arbeit nachvollzogen werden können. Dies gilt für die Leserschaft, aber auch für den Autor dieser Arbeit, welcher mit geringen Vorkenntnissen im Bereich Computer Vision in dieses Projekt startet. Gestützt auf einer Literaturrecherche und Web-Tutorials, werden daher für jeden Prozessschritt die wichtigsten Grundkonzepte erlernt und dokumentiert.

Die erlernte Theorie soll laufend im praktischen Teil umgesetzt werden. Dieser kann in drei Phasen eingeteilt werden. Als erstes wird das Datenset analysiert und vorbereitet. Dies beinhaltet im Wesentlichen das Erstellen von Masken, welche später für das Trainieren und Testen des neuronalen Netzes benötigt werden. Ausserdem müssen die Dateien in ein passendes Format transformiert und anschliessend entweder dem Training-Set oder dem Test-Set zugordnet werden. In Phase Zwei wird das Modell mit den Bildern aus dem Training-Set trainiert. Solange, bis die richtige Parametrisierung gefunden und ein gutes Ergebnis erzielt wird. Danach folgt der Test mit Bildern, welche dem Modell noch nicht bekannt sind. In diesem Schritt werden auch die Vorhersagen gewonnen, welche für die Nachbearbeitung in Phase Drei benötigt werden. Vorhersage heisst in diesem Kontext, dass das Modell Luftbilder analysiert und mit einer bestimmten Wahrscheinlichkeit angibt, ob und wo sich eine Fussgängerinsel befindet. Ist eine Vorhersage positiv, werden die entsprechenden Koordinaten in einer GeoJSON Datei abgespeichert. Aus OSM kann ebenfalls eine GeoJSON Datei exportiert werden, welche alle in der Geodatenbank erfassten Fussgänger Insel einer bestimmten Region enthaltet. Durch einen Vergleich beider Dateien kann überprüft werden, ob OSM gemäss den Luftaufnahmen von Swisstopo auf dem neusten Stand ist.

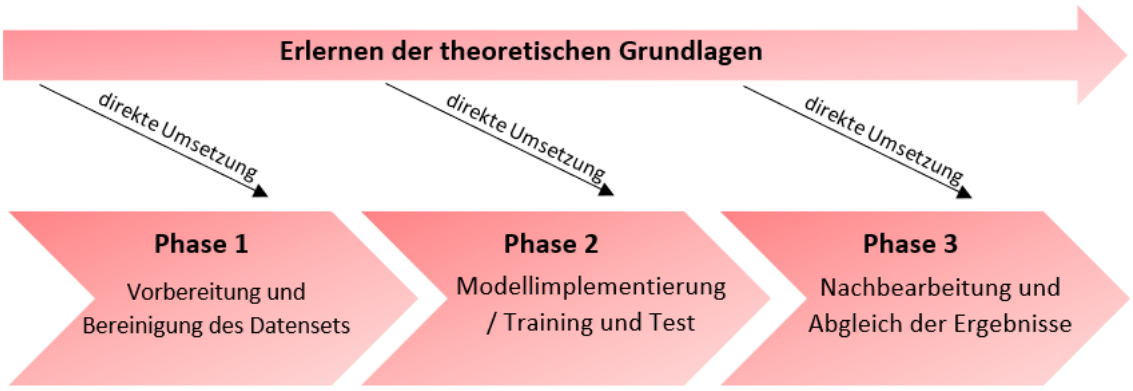


Abbildung 1: Prozessphasen (eigne Darstellung)

1.4.1 Prozessdokumentation

Für die Implementierung des Praxisteils wird die Programmiersprache Python und die web-basierte Umgebung Jupyter Notebook verwendet. Eine Jupyter Notebook Dokument erlaubt es, neben dem Programmiercode, auch eine entsprechende Text-Dokumentation (Markdown) zu erfassen. Ein ergänzende Prozessdokumentation befindet sich daher direkt in diesen Notebooks und ist nicht mit gleichem detaillierungsgrad in diesem Bericht enthalten.

Der Programmiercode und weitere Dokumente sind auf diesem GitHub Repository frei zugänglich: https://github.com/LucienOST/PT1-Refuge_Islands

2 Theoretische Grundlagen

Dieser Kapitel hat das Ziel, die wichtigsten Konzepte, Funktion und Begriffe nochmals aufzufrischen und die Theorie aus der Vorarbeit an einigen Stellen zu ergänzen. Hierbei werden sukzessive die Teilbereiche einer Disziplin beschrieben, die speziell für den praktischen Teil nützlich sind. Auf mathematische, statistische oder informations-technologische Konzepte, welche die Grundlage für diese Disziplin bilden, wird nicht vertieft eingegangen. In Anbetracht der Problemstellung und der Zielsetzung dieses Projektes, ergeben sich für die drei Phasen des Projektes folgende Aufgaben:

Phase 1: Vorbereitung
- Datenbeschaffung
- Erstellen von Masken und Formatierung der Bilder
- Erstellung eines Trainings- und Test-Sets

Phase 2: Implementierung
- Modellbau (bzw. Ausbau) inkl. Hyperparametrisierung
- Model Training
- Model Test und Erstellung von Prognosen

Phase 2: Nachbearbeitung
- Extrahieren und aufbereiten der Prognosen und OSM Daten
- Vergleich der Geokoordinaten

In diesem Kapitel wird versucht Strategien und Zusammenhänge dieser Phasen zu erlernen, damit die jeweiligen Aufgaben anschliessend effizient umgesetzt werden können.

2.1 Grundlagen des Maschinellen Lernens

Maschinelles-Lernen (ML) wird als Teildisziplin der Künstlichen Intelligenz definiert. Es stellt den Ansatz dar, einem Computer das Lernen aus Erfahrung beizubringen. Mit anderen Worten, ein Computer analysiert allgemeine Daten und lernt mit der Zeit Muster und Regeln aus diesen abzuleiten (Inis Ehrlich 2022).

ML kann grundsätzlich in zwei Arten unterteilt werden. Die meist genutzte und für diese Arbeit relevante, ist die des Überwachten Lernens (eng. Supervised Learning). Das Ziel ist es für einen bestimmten Input ($\boldsymbol{x}_i$) denn korrekten Output ($\boldsymbol{y'}_i$) vorauszusagen. Hierbei wird ein ML-Modell mit Daten trainiert, wobei der korrekte Output dem Modell nach einer Prognose bekanntgegeben wird. Somit erhält der Algorithmus direktes Feedback und kann sich somit laufend verbessern. Dieser vorab bekannte Wert wird **Label** ($\mathbf{y}_i$) genannt. Zusammengefasst:

$$D = \{(x_i, y_i)\}^N{}_{i=1}$$

D ist hierbei das Datenset und **N** die Anzahl der Daten. Der Input $\boldsymbol{x}_i$ ist ein D-dimensionaler Vektor und wird Feature oder Attribut genannt. Ein Beispiel zeigt *Abbildung 2*. Die Attribute sind hier die Farbe, Form und ein Boolean-Wert. Das Modell lernt nun alle Eigenschaften von jedem Element im Trainingsset und welcher Klasse das jeweilige Element zugehörig ist. Es wird somit ein Muster gelernt, so, dass wenn dem Modell eine neue Form gezeigt wird, dieses selbständig, und mit Angabe einer bestimmten Wahrscheinlichkeit, angibt welcher Klasse die neue Form angehört (Kevin P. Murphy 2012).

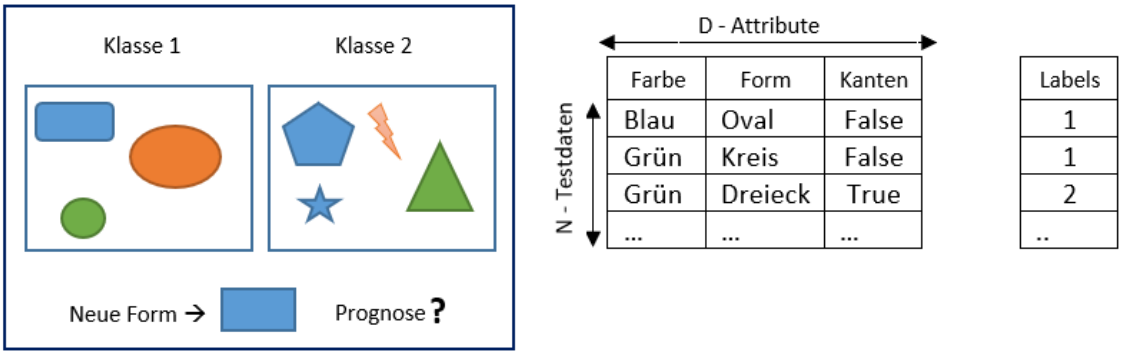


Abbildung 2: Klassifizierungsbeispiel für Maschinelles-Lernen - eigene Darstellung basierend auf (Kevin P. Murphy 2012)

In der Praxis ist ***x*** jedoch meist ein Objekt mit komplexer Struktur, wie zum Beispiel eine Bilddatei oder ein ganzes Textdokument. Der Output ***y'*** kann eine kategoriale oder eine nominale Variable eines begrenzten Sets sein. In diesem Fall wird von einem Klassifizierungsproblem oder von Mustererkennung gesprochen. Ist der Output eine reelle Zahl, handelt es sich um ein Regressionsproblem. Da jedes Input Element x_i mit dem entsprechenden Label ($\mathbf{y}_i$) gemappt ist, kann sofort bestimmt werden, ob eine Vorhersage korrekt war oder nicht -daher der Name «Überwachtes Lernen» (Kevin P. Murphy 2012). Die aktuell relevanteste Form des überwachten Lernens sind neuronale Netze. Diese sind in der Lage, grosse Mengen an unstrukturierten Daten effizient auszuwerten (Inis Ehrlich 2022).

Der wesentliche Unterschied der zweiten ML-Art ist, dass hier die Labels nicht bekannt sind und der Algorithmus somit kein Feedback erhält. Es wird versucht aus einem Datensatz interessante Muster zu finden und unerwartete Erkenntnisse zu gewinnen (eng. knowledge discorvery). Diese Kategorie wird Unüberwachtes Lernen genannt. In diesem Fall ist die Problemstellung vage bis gar nicht definiert und es ist ungewiss ob und welche Resultate am Ende vorliegen (Kevin P. Murphy 2012).

Es existieren Hybridformen dieser zwei Kategorien, in denen dem Modell nur teilweise Rückmeldung gegeben wird. Dies kann hilfreich sein, wenn mit realen Daten aus der Praxis trainiert wird. Oft gibt es dort Grauzonen mit Elementen die noch gar nicht einer Klasse zugeteilt werden konnten. Ein weiterer Spezialfall, welcher oft auch als dritte ML-Kategorie bezeichnet wird, ist das Bestärkende Lernen (eng. reinforcement learning). Hierbei besteht das Feedback nicht nur aus Richtig oder Falsch. Prognostiziert ein Algorithmus einen inkorrekten Output, wird er bestraft. Handkehrum, gibt es eine Belohnung bei korrekten Vorhersagen (Daniel Trabold 2021).

2.1.1 Trainings- und Testdaten

Wie erfolgreich ein ML-Modell seine Aufgabe löst, hängt in erster Linie von der **Qualität der Daten** ab, mit denen es trainiert wird. Daten werden in der Vorbereitungsphase neu erstellt, gesammelt und zu einem Set zusammengetragen. Sie müssen den **Kontext** der Problemstellung in adäquater und generalisierter Art und Weise widerspiegeln. Nur so kann ein Algorithmus lernen, ein Problem allgemein und ohne Voreingenommenheit (**Bias**) zu lösen. Wie im vorherigen Kapitel beschrieben, wird beim Überwachten Lernen jedem Datenelement ein Label zugewiesen. Diese müssen manuell erstellt werden. Sind sie fehlerhaft, scheitert auch das Modell (Jared Lander und Michael Beigelmacher 2020).

Ist ein Datenset erstmal erstellt, heisst es noch nicht, dass es direkt einsatzbereit ist. Oft müssen Daten normalisiert, formatiert oder bereinigt werden. Anschliessend werden die enthaltenen Elemente je einem Trainings- und einem Test-Set zugewiesen. Niemals sollte ein Modell mit dem ganzen Datenset trainiert werden da ansonsten dessen Qualität nicht überprüft werden kann. Mit dem Trainings-Set lernt ein Algorithmus ein Problem selbständig zu lösen. Ist dieser nicht fähig mit den vorhandenen Trainingsdaten einen Lösungsweg zu erlernen, wird von **Underfitting** gesprochen. Das heisst, dass die Architektur oder die Parameter des Modelles überarbeitet werden müssen. Performt hingegen ein Modell im Trainingsprozess sehr gut, versagt aber, sobald neue Daten aus der gleichen Problemdomäne gezeigt werden, wird von **Overfitting** gesprochen. Das heisst das Modell hat die spezifischen Eigenschaften des Trainingssets so stark verinnerlicht, dass es das Problem nicht mehr allgemein lösen kann. Diese Überanpassung kann durch **Ausreiser** oder **Bias** in den Trainingsdaten verursacht werden. Um dem vorzubeugen, braucht es das Test-Set. Diese besteht aus Daten, die dem Modell noch nicht gezeigt wurden und kann somit dessen Performance evaluieren (Helmut Grabner und Christoph Würsch 2021).

Es existieren zahlreiche Metriken für die Evaluierung eines Modells. Zu den wichtigsten gehören Precision, Recall, F1-Score, welche auch im Praxisteil dieser Arbeit verwendet und an entsprechender Stelle genauer erklärt werden.

2.2 Computer Vision

Computer Vision ist ein Teilbereich des maschinellen Lernens welcher sich mit der Extraktion von Semantik aus bildhaften Daten der realen Welt befasst. Computer Vision orientiert sich an den Strukturen und Eigenschaften der dreidimensionalen Welt. Im Prinzip wird versucht, menschliche Sehvorgänge mit Hilfe von neuronalen Netzen digital nachzubilden. In der Analyse der Bilder und Videos werden die **geometrischen Eigenschaften** (Formen, Grössen und Lage), die **Materialeigenschaft** (Farbe, Oberflächen und Texturen) und die **Beleuchtung** (helle oder dunkle Regionen) miteinbezogen. Zu den Hauptaufgaben von Computer Vision gehören die Objekterkennung, sowie die Segmentierung und Semantisierung dieser Objekte. Die Semantisierung kann die blosse Zuteilung eines Objektes zu einer Klasse sein, oder aber auch das Interpretieren von ganzen Bewegungen (R. Sablatnig und S. Zambanini 2021).

Ein digitales Bild ist im Grunde nichts anderes als eine **Matrix** aus **Bildpunkten**, welche je einen bestimmten Wert auf einer Skala einnehmen können. Computer interpretieren Bilder anhand dieser Werte, wobei jeder einzelne Pixelwert als Input-Wert gewertet wird (Alexander Amini 2020). Grauton-Bilder können beispielsweise durch eine zweidimensionale Matrix abgebildet werden:

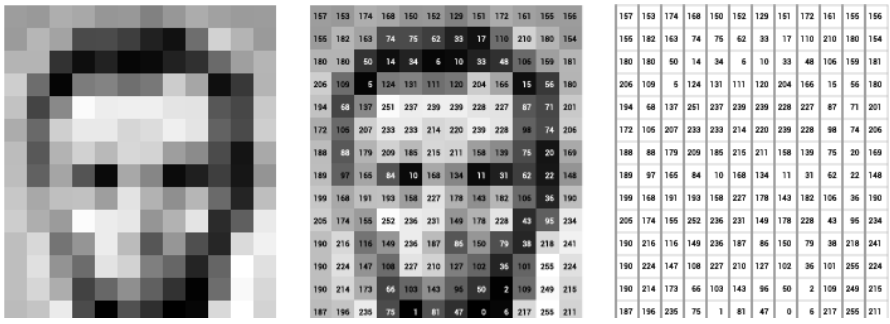


Abbildung 3: Pixelmatrix zur Bildanalyse (Alexander Amini 2020)

In diesem Beispiel nimmt jeder Pixel einen Wert zwischen 0 (schwarz) und 255 (weiss) ein. Sprich, tiefe Werte repräsentieren dunkle Regionen und Werte nahe 255 helle Regionen. Das Bild besteht aus 12 Spalten und 16 Reihen, was insgesamt 192 Input Werte für die Analyse dieses Bildes bedeutet. Bei Farbbildern wird die Matrix um eine weitere **Dimension** erhöht -je einem Wert für die drei Grundfarben Rot, Grün und Blau (RGB). Für eine Farbbild im gleichen Format bedeutet dies bereits 576 Input Werte. ML-Modelle -insbesondere im Bereich Deep Learning- werden mit tausenden von Bildern trainiert. Für jedes Bild in einem Format von 1024 x 768 werden über 2 Megabytes Speicher benötigt (8 Bit pro Farbwert). Die Bildanalyse mittels Computer Vision ist also eine sehr **Memory** lastige Aufgabe, die Rechenleistung und Zeit beansprucht (Data Robot 2018).

Eine Lösung für dieses Problem besteht darin die Bilder vor dem Training des Modelles zu verkleinern. Dabei gilt es darauf zu achten, dass die grundlegenden Eigenschaften der Objekte und Muster, die in den Bildern abgebildet sind, beizubehalten. Die dafür angewendete mathematische Operation wird **Interpolation** genannt (Raghul Asokan 2021). Auch das ursprüngliche Format (Quer, Längs oder quadratisch) sollte beibehalten werden, damit keine Information verlorengehen -zum Beispiel durch Verzerrung. In welchem Ausmass abwärts skaliert werden kann, hängt vor allem von der Aufgabe ab, welches das Modell erfüllen soll. Müssen beispielsweise kleine, eher dezente Objekte im Bild erkannt werden, gehen bei zu starker Verkleinerung wichtige Details verloren (Joseph Nelson 2020).

2.2.1 Semantische Bildsegmentierung

Bildsegmentierung ist wiederum ein Teilbereich von Computer Vision. Semantische Segmentierung in diesem Kontext bedeutet, jeden einzelnen Pixel eines Bildes zu überprüfen, um festzustellen, ob dieser einem gesuchten Objekt zugehörig ist. Das Ziel ist es homogene Bildpunkte zu einer Gruppe zusammenzufassen und diese Gruppen anschliessend sinnvollen Objekten zuordnen. Die Objekte werden hierfür einerseits vom Hintergrund getrennt und andererseits auch voneinander. Es existieren zahlreiche ML-Verfahren, die zur Lösung einer solchen Aufgabe genutzt werden können. Das Grundkonzept besteht darin, Änderungen der Pixelwerte eines Bildes in Abhängigkeit ihrer **Pixelposition** zu untersuchen. Je nach Methode werden einzelne Pixel oder ganze Regionen -auch Cluster genannt- untersucht. Am Schluss ist jedem Pixel eine Objektklasse zugeteilt. Ist das Ziel der Bildsegmentierung die Abtrennung eines **einzigen** Objektes -wie zum Beispiel Fussgängerinseln in Orthofotos- ist das Ergebnis ein **Binär-Bild**. Die Pixel in der Matrix, die dem gesuchten Objekt angehören erhalten also den Wert 1 (weiss). Alle anderen Werte sind 0 (Jens Kürbig und Martina Sauter 2006).

Diese Binär-Dateien werden **Masken** genannt. Diese sind in zweierlei Hinsicht von Bedeutung. Binär-Masken bilden einerseits die Labels, mit denen eine Algorithmus trainiert (eng. **Target Masks**). Sind sie ein wesentlicher Bestandteil des Datensets, welches für eine ML-Projekt vorbereitet werden muss. Bei der Erstellung eines Datensets aus Bildern, muss für jedes Bild manuell vermerkt werden, wo sich das Zielobjekt befindet. Dies wird Polygon-Annotation genannt. Gespeichert werden hierbei die Randpunkte des Segments, in Form von Polygon-Koordinaten. Alle Punkte, die sich innerhalb dieser Region befinden, werden anschliessend dem entsprechenden Label zugewiesen. Im Normalfall enthält ein Datenset für einen Segmentierungs-Algorithmus mehrere tausend Bilder. Die Vorbereitungsphase für eine solche Aufgabe ist daher äusserst zweitaufwendig. Mittlerweile existieren jedoch auch semi-automatische Annotations-Tools, die das Markieren der Segmente stark beschleunigen. Manuell muss hierbei nur noch grob eine Bounding-Box in der Zielregion einzeichnen werden, die detaillierte Einzeichnung übernimmt anschliessend das Tool selbst (Hmrishav 2022).

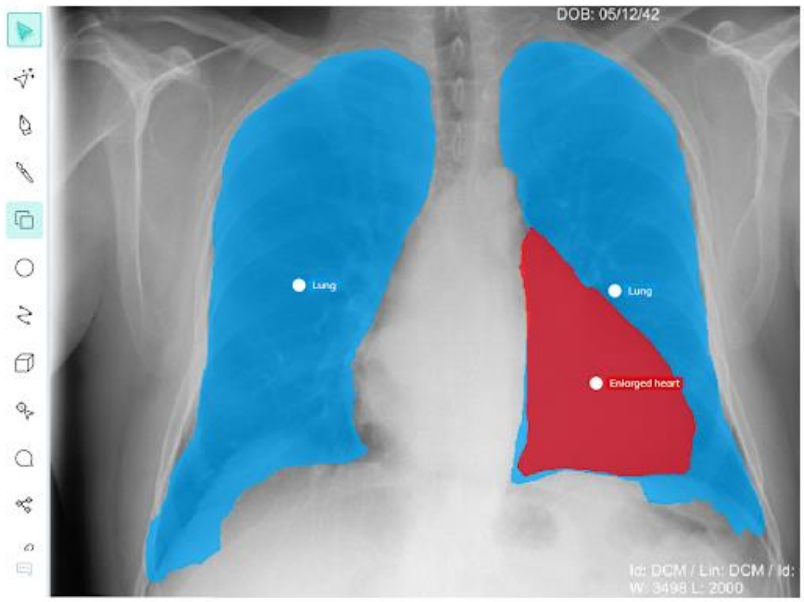


Abbildung 4: Segmentierung von Objektklassen durch das Bildbearbeitungstool V7 (Hmrishav 2022)

In der Bildsegmentierung fungieren Masken aber nicht nur als Labels. Auch der Output eines Modells sind binäre Masken (eng. **Predicted Masks**). Sprich, hier erstellt der Algorithmus diese selbst. Nämlich dort, wo dieser das Objekt vermutet, welches gemäss seiner Aufgabe segmentiert werden soll (Rachel Zheng 2020).

2.3 Neuronale Netze

Aufgrund der Verwendung eines künstlichen neuronalen Netzes im praktischen Teil dieser Arbeit, werden in diesem Kapitel die Grundkonzepte erläutert. Konkret wird in diesem Projekt ein Convolutional Neural Network (CNN) genutzt, welches in einem separaten Kapitel genauer erläutert wird.

2.3.1 Herkunft und Inspiration

Das Konzept der künstlichen neuronalen Netze leitet sich vom menschlichen Denkprozess ab. Der Begriff Neuron stammt aus der Biologie und ist eine Nervenzelle im Gehirn, welche sich mit Milliarden anderer Neuronen zu einem Netz zusammenschliesst und dort elektrische Signale austauschen kann (Inis Ehrlich 2022). Der Mensch ist sehr gut darin verschiedenste Objekte, Formen oder Zeichen innert kürzester Zeit zu erkennen, zuzuordnen und zu unterscheiden. Gemäss neurowissenschaftlichen Erkenntnissen erfolgt die dazu notwendige Verarbeitung im Gehirn durch das Aussenden elektrischer Impulse bestimmter Neuronen. Entscheidend ist hierbei die Stärke des Signals, denn bei Unsicherheit können auch mehrere Neuronen gleichzeitige feuern. Die Idee von **künstlichen** Neuronalen Netzen ist es diesen Prozess digital nachzubilden (Helmut Grabner und Christoph Würsch 2021).

2.3.2 Aufbau und Funktion eines Neuronalen Netzes

Ein Neuronales Netz besteht aus drei Ebenen. Denn **Rezeptoren** für den Input. Diese können Bilder, Töne, Texte oder andere Signale entgegennehmen. Einer Zwischenebene die aus verschiedenen Schichten (**Layers**) mit Neuronen (**Nodes**) besteht. In jeder dieser Schichten werden Daten transformiert und Berechnungen durchgeführt. Die Berechnungen eines Neurons werden an verschiedene Neuronen der nachfolgenden Schicht weitergeben. Die letzte Ebene bildet der **Output**, welcher die Endergebnisse in Form einer Prognose liefert (Inis Ehrlich 2022). Zur Veranschaulichung ein einfaches Beispiel:

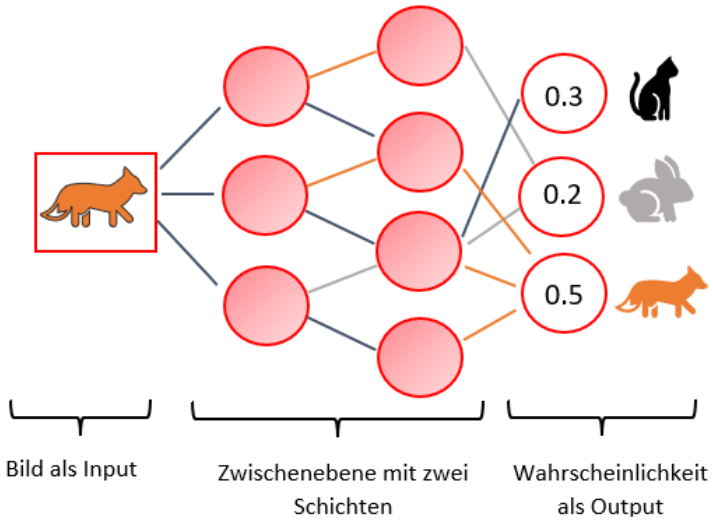


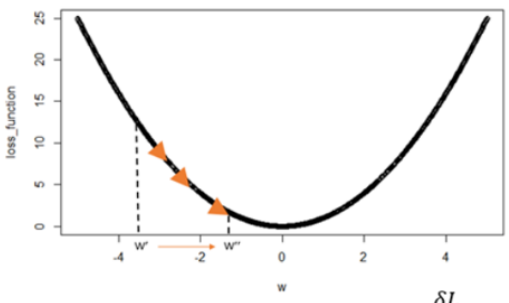
Abbildung 5: Konzept eines neuronalen Netzes - eigene Darstellung basierend auf (Inis Ehrlich 2022)

Der entscheidende Prozess findet in der Zwischenebene statt. Jede einzelne Schicht fungiert hierbei als eigner Algorithmus. Anders gesagt, wird in jeder Schicht eine unterschiedliche Methode angewandt, um die Eigenschaften der Inputdatei zu erlernen.

Ist der Input ein Bild, fokussiert sich beispielsweise ein Layer auf kantige Formen, eine weiterer analysiert die Farbveränderungen. In einer Schicht werden einzelne Pixel untersucht, in anderen ganze Bildregionen. Diese Fähigkeit verschiedene Ansätze zu kombinieren und deren jeweiligen Vorteile zu nutzen, ist eine der Stärken von neuronalen Netzen. Die Anzahl Layers und Nodes sind die **Hyperparameter** des Modelles und werden vom Nutzer bzw. der Nutzerin und nicht vom Modell selbst bestimmt. Im Prinzip können beliebig viele Schichten mit jeweils beliebig vielen Neuronen aneinandergereiht werden. Dabei gilt es aber zu beachten, dass die Anzahl Berechnung bei jeder Erhöhung exponentiell steigen kann. Dementsprechend mehr Rechenleistung wird benötigt und desto länger dauert ein Trainingsdurchlauf (Helmut Grabner und Christoph Würsch 2021).

Künstliche Neuronale Netze zeichnen sich auch dadurch aus, dass sie sich selbständig verbessern. Hierfür wird in den meisten Fällen das Gradientenabstiegsverfahren verwendet (**Backpropagation)**. Bei einem untrainierten Modell müssen die **Parameter** für die Berechnungen zu Beginn initialisiert werden. Im Verlaufe des Trainings werden diese laufend vom Modell angepasst, bis die gewünschten Ergebnisse erzielt werden. Parameter in einem Neuronalen Netzwerk sind typischerweise **Gewichtungen** von Neuronen. Ein Neuron mit hohem Gewicht bedeutet, dass dessen Berechnung mehr Einfluss hat als die von anderen. Mit der Zeit lernt das Modell, welche Schicht und welche Neuronen für die Lösung des ihm aufgetragenen Problems nützlicher sind als Andere und optimiert sich somit selbst (Alessandro Giusti 2021).

Die Initialisierung und Optimierung dieser Koeffizienten erfolgten nicht willkürlich, sondern basiert auf einer vorab definierten **Strategie**. Die Parameter werden durch die **Aktivierungsfunktion** initialisiert. Je nach Netztopologie und Problemstellung eignen sich verschiedene Funktionstypen. Ihre Aufgabe ist es, die Eingabewerte in den verwendbaren Output umzuwandeln. Bei einem Wahrscheinlichkeitswert als Output muss beispielsweise ein Wert zwischen 0 und 1 herauskommen. Verschiedene Aktivierungsfunktionen können in einem Neuronalen Netz kombiniert und an beliebiger Stelle plaziert werden. Die Optimierungsstrategie wird durch die Verlustfunktion und **Lernrate** definiert. Die Lernrate entscheidet darüber, wie stark Gewichtungen nach einem Fehler angepasst werden, was wiederum die **Verlustfunktion** beeinflusst. Das Ziel der Verlustfunktion ist immer die **Minimierung** der Modell-Fehlerrate. Hierfür muss auf dem Funktionsgraphen ein globales Minimum gefunden werden. Das Gradientenabstiegsverfahren ist ein Algorithmus zur Lösung dieser Aufgabe (Valentina Alto 2019). Ein eindimensionales Beispiel zeigt *Abbildung 6.*



$$w_i' = w_i - \frac{\delta L}{\delta w_i}$$

Wobei: W_i = akutelles Gewicht, $\frac{\delta L}{\delta w_i}$ = Gradient der Verlustfunktion

Abbildung 6: Optimierung der Lernrate (Valentina Alto 2019)

Es gibt verschiedene Arten von Verlustfunktionen, die wiederum für verschiedene Anwendungsfälle geeignet sind. Sie sind Indikatoren dafür, wie gut ein Trainingsdurchlauf mit der momentanen Parametrisierung abschneidet. Durch sie lernt ein Modell, wie die Parameter am besten anzupassen sind. Dabei muss auf eine adäquate Regularisierung geachtete werden. Das heisst, es muss verhindert werden, dass einzelne Neuronen zu hoch oder zu tief gewichtet werden. Dies ist nötig, da ein ML-Modell einen allgemein anwendbaren Ansatz erlernen soll, um ein Problem zu lösen und sich nicht an den Ausreiser des Training-Sets orientiert (Artem Oppermann 2021). Zusammengefasst nochmals der Ablauf in einem neuronalen Netz:

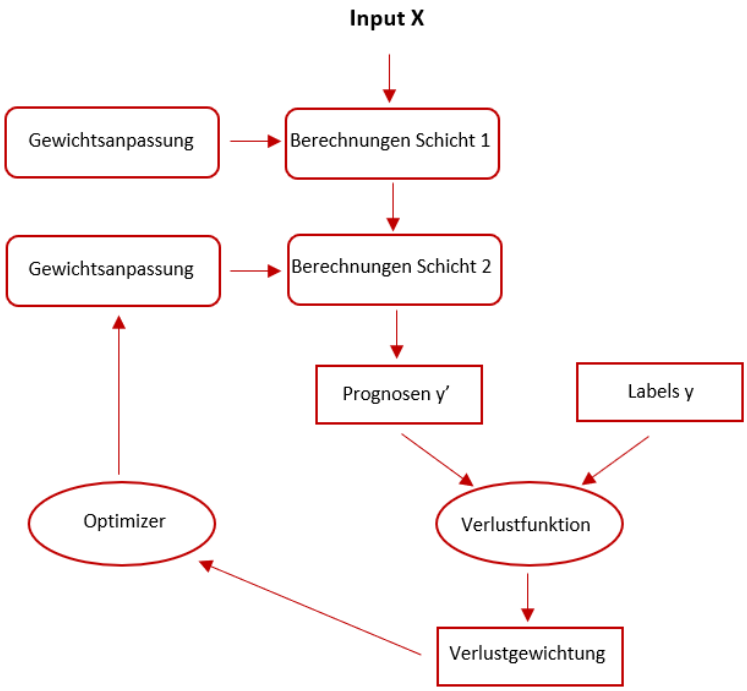


Abbildung 7: Lernprozess eines neuronalen Netzes - eigene Darstellung

2.3.3 Vorbereitung

Ein wichtiger Schritt vor der Nutzung eines Neuronalen Netzes, ist die **Normalisierung** der Input-Daten. Bei den meisten Neuronalen Netze wird vorausgesetzt, dass alle Daten die gleiche Grösse haben. Handelt es sich um Bilder, heisst Normalisierung also, dass alle Bilder ins gleiche Format konvertiert werden müssen. Wird mit grossen Datensets trainiert, ist es ausserdem nicht effizient alle Daten gleichzeitig während einem Trainingsdurchlauf zu nutzen. Statt dessen werden weitere kleinere Sets (**Batches**) erstellt und dem Modell sukzessive eingespeist (Valentina Alto 2019).

2.3.4 Deep Learning

Deep Learning und Neuronale Netze werden oft synonym verwendet. Nicht jedes Netz nutzt jedoch einen Deep Learning Ansatz und nicht jedes Deep Learning Modell ist ein neuronales Netz. Der Unterschied beim Deep Learning ist, dass das Modell selbständig Regeln und Lösungsansätze aufstellt, welche den Programmierer oder der Programmiererin nicht bekannt sind. Zum Vergleich, bei herkömmlichen ML-Ansatz muss stets ein klarer Ablauf, ein Algorithmus, definiert werden, welcher das Problem lösen soll.

Bei einem Deep Neural Network ist nicht bekannt welche Input-Attribute analysiert werden. Daher wird von versteckten Schichten gesprochen (Hidden Layers). Speziell ist, dass beim Deep Learning auch Probleme gelöst werden können, für die das Modell ursprünglich gar nicht trainiert wurde. Aufgrund dieser Eigendynamik ist Deep Learning ein Spezialfall im Bereich des überwachten Lernens. Das bekannteste Deep Learning Modell ist das Convolutional Neural Network, welches sich insbesondere für Computer Vision eignet (Yulia Gavrilova 2020).

2.4 Convolution Neural Networks

Convolutional Neural Networks (CNN) sind sehr gut darin Muster auf verschiedenen Tiefeneben eines Bildes zu erkennen. Sie wenden hierfür eine mathematische Operation an die Convolution genannt wird. Sie unterscheiden sich von herkömmlichen Neuronalen Netzen durch ihre Architektur, die sich aus drei Arten von Schichten zusammensetzt.

2.4.1 Convolution

Aus mathematischer Perspektive ist Convolution das Produkt von zwei Funktionen. Das Ergebnis dieser Operation liefert wiederum eine dritte Funktion. Im Kontext der Bildanalyse kann diese Operation als eine Art Filter betrachtet werden, wobei als Ergebnis ein neues Bild herauskommt. Dieser Filter wird **Kernel** genannt. Ein Beispiel: Die erste Funktion **g(x)** ist das ursprüngliche Bild. Wie in Kapitel 2.2 erwähnt, interpretiert ein Computer eine Bilddatei als **Zahlenmatrix**. Der Kernel als zweite Funktion **f(x)** ist ebenfalls eine Matrix. Während der Convolution iteriert nun der Kernel über jeden Pixel von f(x) und kreiert dabei erneut eine Matrix **g(x)*f(x)**. Diese stellt ein neues Bild dar (Grant Sanderson 2020).

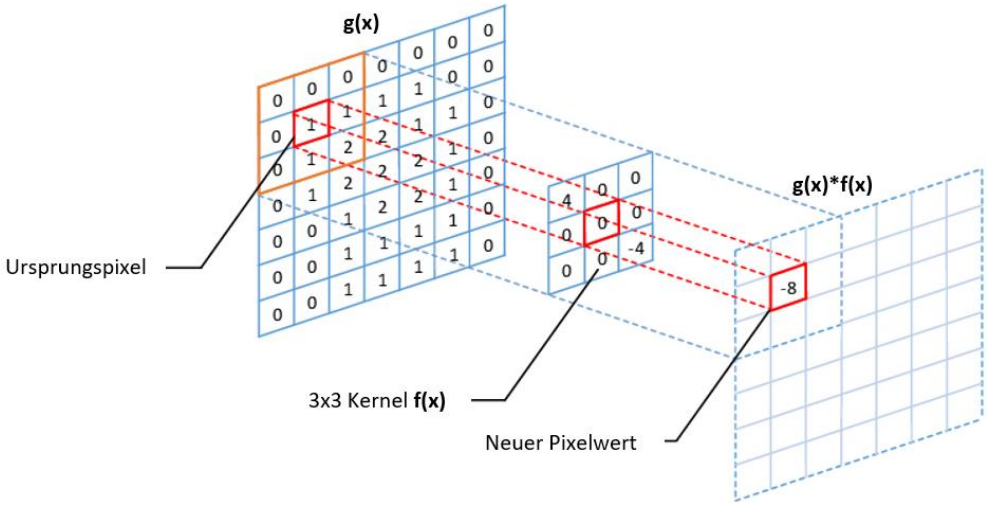


Abbildung 8: Convolution von Pixelwerten - eigene Darstellung basierend auf (Zhao Guyu et al. 2019)

Durch die Convolution können Konturen und Muster in Bilder hervorgehoben werden. In einem CNN führt dies dazu, dass das Modell nicht nur das ursprüngliche Bild analysiert, sondern auch leicht abgeänderte Versionen davon. Der Kernel kann unterschiedlich gross sein und wird je nach Art der Filterung mit unterschiedlichen Werten gefüllt. Der Gausssche Kernel wird beispielsweise zur Weichzeichnung eines Bildes verwendet. Der Name beruht auf der Gaussschen Normalverteilung. Je nach Grösse des Kernel verstärkt sich der Effekt (Alessandro Giusti 2021).

Abbildung 9: Kerneleffekt bei unterschiedlicher Kernelgrösse (Alessandro Giusti 2021)

2.4.2 Architektur

Ein CNN setzt sich normalerweise aus drei verschieden Arten von Schichten zusammen: Convolutional Layers, Pooling Layers und Fully-Connected Layers (Keiron O'Shea und Ryan Nash 2015). Die Architektur des Netzes lässt sich zu dem in zwei Teile aufteilen: Einem **Encoder**, welcher für das Erlernen der Features den Input transformiert, und einem **Decoder**, der für die Rekonstruierung der Eingabedatei und dessen Klassifizierung sorgt (Hmrishav 2022). Mit anderen Worten, der Input in ein CNN wird zuerst schrittweise verkleinert (**downsampling**) und anschliessend wieder schrittweise vergrössert (**upsampling**) (Thomas Koller 2021).

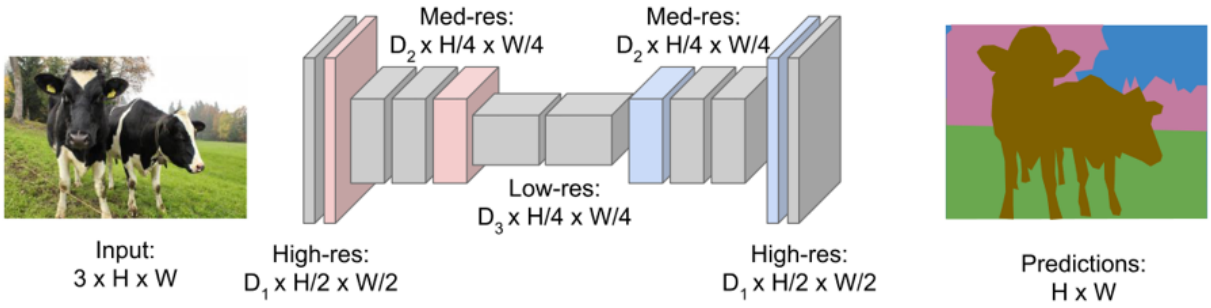


Abbildung 10: Schichten eines Convolutional Neural Networks (Thomas Koller 2021)

Die drei Schichtarten erfüllen folgende Aufgaben:

1. **Convolutional Layer**
   In dieser Schicht werden die zuvor erläuterten Kernels eingesetzt. Es werden also Informationen aus Input extrahiert und umgewandelt. Das Modell lernt durch diese Kernels und bestimmte Neuronen «feuern» wenn ein spezifisches Feature an einer bestimmten Stelle im Raster erkannt wird (Keiron O'Shea und Ryan Nash 2015). Idealerweise werden mehrere Convolutional Layer in einem Netz plaziert, damit verschiedene Kernelvariationen in verschiedenen Grössen kombiniert werden können. Dadurch können mehr Informationen aus den Bildern gewonnen werden (Alessandro Giusti 2021).

2. Pooling Layer

Pooling Layer haben einerseits die Aufgabe die Dimensionalität des Input Vektors zu verkleinern. Dies soll die Anzahl Parameter reduzieren und damit auch die Komplexität der Berechnungen im Modell (Keiron O'Shea und Ryan Nash 2015). Durch diese Operation geht jedoch auch Information verloren. Daher findet diese Verkleinerung in schrittweise und in Abwechslung mit einem Convolutional Layer statt. Es gibt zwei Ansätze: Bei der **Max Pooling** Methode wird der grösste Pixelwert einer Region weitergegeben, bei der **Average Pooling** Methode wird zuerst der Durchschnittswert der Region berechnet (Thomas Koller 2021).

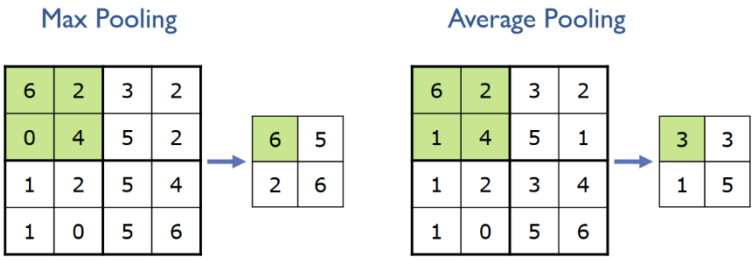


Abbildung 11: Max Pooling vs. Average Pooling (Thomas Koller 2021)

3. Fully-Connected Layer

Diese Schichten kommen erst am Ende des Netzes zum Einsatz. Es ist die Art von Layer, welche sich auch in herkömmlichen neuronalen Netzen befinden. Sie sind vereinfach gesagt dafür zuständig aus den extrahierten und transformierten Informationen durch die vorherigen Schichten, die richtigen Schlüsse zu ziehen. Durch sie wird auch bestimmt in welcher Form der Output ausgelesen wird (Keiron O'Shea und Ryan Nash 2015).

Die Zusammensetzung eines neuronalen Netzes aus verschiedenen Schichten wird Architektur des Modells genannt. Prinzipiell können diese Schichten in beliebiger Reihenfolge und Anzahl zu einem Netz zusammengefügt werden. In der Praxis wird jedoch vermehrt auf bewährte Architekturen zurückgegriffen. Für die semantische Bildsegmentierung hat sich unter anderem die Architektur des **Residual Networks** als geeignet erwiesen (Thomas Koller 2021).

2.5 Geoinformationssystem

Neben der Domäne des Maschinellen Lernens werden in dieser Arbeit auch Themen aus dem Bereich Geoinformationssystem (GIS) behandelt. In diesem Kapitel sollen daher auch in diesem Kontext nützliche Konzepte erläutert werden. Die hierbei erlernten Grundlagen sind vor allem für die Phase der Nachbearbeitung relevant.

2.5.1 Maschinelles Lernen und GIS

Das Potential von Künstlicher Intelligenz wurde längst auch für den GIS Bereich erkannt. Eine realitätsnahe und aktuelle Karte der Welt digital abzubilden, benötigt enorme Datenmengen. Solche Kartendienste existieren bereits (zum Beispiel Google Maps, OpenStreetMap u.a.). Doch die Welt ändert sich ständig. Der Mensch baut Städte und Dörfer aus, baut neue Verkehrslinien und Wanderwege, staut Flüsse oder leitet diese um.Natürlich kann auch die Natur selbst, die Topologie der Erde verändern.

Die grosse Herausforderung für diese Kartendienste ist es, diese Änderungen zeitnah und exakt in ihren bestehenden Systemen zu übernehmen. Eine Möglichkeit wie Karten auf Aktualität überprüft werden können, ist durch die Überprüfung von Luftbildern. Manuell durchgeführt, ist dies jedoch ein sehr mühseliger und zeitaufwendiger Prozess. Verschiedene Infrastruktur muss hierbei spezifisch markiert und klassiert werden, bevor aus solchen Luftbildern aussagekräftige Karten werden (Thomas Paschke 2020). Wie in das vorherige Kapitel erwähnt, ist ein ML-Modell sehr effizient darin grosse Datenmengen zu analysieren. Gerade die Methoden der Semantischen Bildsegmentierung eigen sich sehr gut dafür, diesen Überprüfungsprozess in Luftbilder zu übernehmen oder zumindest zu unterstützen.

2.5.2 Geodaten

Gemäss (Swisstopo (a) 2020), sind Geodaten digitale Informationen, denen auf der Erdoberfläche eine bestimmte räumliche Lage zugewiesen werden kann. Sei dies in der Form von Koordinaten, Ortsnamen, Postadressen oder andere Kriterien. Zusammen liefern diese Daten die Grundlage für unterschiedliche Produkte wie zum Beispiel Karten- und Navigationssysteme.

Bei der Erstellung dieser Produkte spielen digitale Luftbilder eine wichtige Rolle. Orthofotos ermöglichen eine verzerrungsfreie und massstabgetraue Abbildung der Erdoberfläche. Zusammen bilden sie eine Orthofotokarte, auch Orthomosaik-Karte genannt. Orthofotos liefern einerseits solchen Karten den grafischen Hintergrund. Andererseits, enthalten diese Bilddateien weit mehr Informationen als eine blosse Matrix aus Pixelwerten. In der Datei befinden sich direkt auch ein Koordinatengitter, welches die Zuweisung von Informationen an die jeweiligen Bildpunkte ermöglicht (Hans-Peter Bähr 2005). Die Zuweisung von raumbezogener Information zu einem Datensatz wird Georeferenzierung genannt. Handelt es sich um geopraphische Koordinaten, wird auch von Geokodierung oder Geotagging gesprochen. Damit dies möglich ist, wird ein entsprechendes Koordinatenreferenzsystem (eng. coordinate reference system –**CRS**) vorausgesetzt (Michael Herter 2008).

Je nach Anwendungsfall, eignen sich unterschiedliche CRS Arten. Diese unterscheiden sich teilweise durch ihre mathematischen Ansätze zur Erstellung des Koordinatensystems sowie der Form der Koordinaten. Auch muss beachtet werden, dass bestimmte Systeme lokal begrenz sind -zum Beispiel nur innerhalb eines Landes anwendbar sind. Zur Identifizierung hat jede CRS Art einen **EPSG Code** zugewiesen. Dieser besteht jeweils aus vier oder fünf Zeichen. Für die Lokalisierung von Punkten auf der ganzen Erde sind geographische Koordinatensystem geeignet, genannt **WSG84** (EPSG Code: 4326). Dieses bildet eine dreidimensionale, sphäroidische Oberfläche, auf welcher jeder Punkt der Erde durch die Angabe von **Längen-** und **Breitengraden** referenziert werden kann (Leah Wasser 2019).

Das für die Schweiz relevante Bezugsystem trägt den Namen **LV95** (EPSG Code: 2056). Das Bundesamt für Landestopografie (Swisstopo), welches für die Orthokarte der Schweiz zuständig ist, referenziert ihre Orthofotos anhand dieses Koordinatensystems (Swisstopo (b) - Kein Datum).

3 Werkzeuge

Für die technische Umsetzung dieses Projektes wird eine Reihe von Tools und Formaten genutzt. Die wichtigsten sollen in diesem Kapitel aufgelistet werden. Deren Auswahl beruht nicht auf einem vertieften Selektionsverfahren. Es werden grundsätzlich dieselben Werkzeuge verwendet, wie in der Vorarbeit. Es gilt darauf hinzuweisen, dass es im Bereich Computer Vision und Bildsegmentierung andere mögliche ML-Ansätze, andere Programmiersprachen oder -bibliotheken die gleichen oder sogar bessere Ergebnisse erzielen könnten.

Programmier-umgebung

Der gesamte Programmcode des Praxisteils mit der Programmiersprache **Python Version 3.9** geschrieben. Genutzte Pakete werden Mithilfe der Distribution **Anaconda** in einer virtuellen Entwicklungsumgebung verwaltet. Der Programmiercode und die dazugehörige Dokumentation sind in **Jupyter Notebooks** (.ipynb Files) gespeichert.

Python Framework

Fastai ist die zentrale Python Bibliothek, die für dieses Projekt verwendet wurde. Sie fungiert als Baukasten für Deep Learning Modelle, mit der Nutzerinnen und Nutzer vorgefertigte High-Level Komponenten von neuronalen Netzen für ihr eigenes Modell verwenden können. Ein weiterer Vorteil ist, dass über die Fastai API bereits vortrainierte Modelle geladen werden können. Zum Beispiel kann ein Convolutional Neural Network weiterverwendet werden, welches auf das Erkennen von Ecken und Kanten bereits vortrainiert ist. Dieser Teil des Trainingsprozesses muss daher nicht mehr selbst durchgeführt werden. Dies vereinfacht den Prozess und spart Zeit. Fastai basiert auf dem etablierten Deep Learning Framework **PyTroch** von Facebook auf. Fastai abstrahiert den Prozess des Modellbaus aber noch weiter, bietet dafür aber weniger Konfigurationsmöglichkeiten (fast.ai 2022). Die korrekte Anwendung von fastai wurde im Online Kurs «Practival Deep Learning for Coders» erlernt. Link zum Kurs: https://course.fast.ai/

Für den Bereich Bildverarbeitung und Computer Vision werden die freien Python Bibliotheken **OpenCV** und **Scikit-Image** verwendet. OpenCV wurde von Intel entwickelt und wird heute von einem Subunternehmen gepflegt. Der Code ist OpenSource. Für die Bildbearbeitung steht eine Vielzahl an Algorithmen zur Auswahl. Diese sind insbesondere für die Vorbereitungsphase und die Nachbearbeitung in einem ML-Projekt hilfreich und zeichnen sich durch ihre Geschwindigkeit aus (Bradski und Kaehler 2011). In diesem Projekt wird OpenCV sowohl für das Erstellen der Labels wie auch für die Generierung von Prognosen verwendet. Scikit-Image bieten ebenfalls eine Auswahl an Methoden für die Bildbearbeitung an. Für den Praxisteil wird Scikit-Image für das Transformieren der Bilder in das richtige Format genutzt.

Für Nutzung von raumbezogenen Rasterdaten wird **Rasterio** verwendet. Diese Bibliothek wird für die Nachbearbeitung benötigt, um aus den Orthofotos Koordinaten zu extrahieren und diese anschliessend in das Format des gewünschten Koordinatenreferenzsystems umzuwandeln.

Für die praktische Umsetzung werden noch weiter Bibliotheken von Python genutzt, welche für kleinere Unterstützungsaufgaben benötigt werden. Diese werden Aufgrund ihrer allgemeinen Anwendbarkeit nicht weiter beleuchtet.

Dateienformate für Geodaten

Die Orthofotos von Swisstopo werden als **Geo-TIF** Dateien (Tagged Image File Format – .tif) bereitgestellt. TIF-Dateien sind in erster Linie Bilder, die in einem hochwertigen Grafikformat gespeichert werden können. Sie bieten jedoch auch die Möglichkeit **geographische** und **kartographische** Informationen in der Form von **Tags** in die Bild-Datei einzubetten. Eine Geo-Tif Datei weicht dabei nicht vom herkömmlichen TIF-Format ab (datei.wiki).

GeoJSON ist ein offener Standard um geographische Daten, wie beispielsweise Längen und Breitengrade, zusammen mit anderen, nicht räumlich bezogenen Attributen zu repräsentieren. Die Struktur basiert auf der JavaScript Object Notation (JSON). Geometrien können in der Form von Punkten, Linien oder Polygonen erfasst werden. In diesem Projekt wird ein GeoJSON verwendet, um die vom Modell prognostizierten Koordinaten zu speichern und mit einem zweiten GeoJSON mit Daten aus OpenStreetMap abzugleichen (geojson.org).

OpenStreetMap

OpenStreetMap (**OSM**) ist ein Gemeinschaftsprojekt mit dem Ziel eine akkurate Karte der Welt zu erstellen. Freiwillige Nutzerinnen und Nutzer können ihr lokales Wissen nutzten und selbst Beitrage erfassen und pflegen. Die Webseite und die zugrundeliegenden Dienste werden von der OSM-Foundation betrieben. Mit einem entsprechenden Vermerk können die Informationen von OSM von jedem und jeder für beliebig Zwecke kostenlos verwenden (OSM Org. 2022). Durch den Open-Data Ansatz und durch den einfachen Zugang eignet sich dieses Tool besonders auch für den akademischen Bereich.

Um Daten aus OSM programmiertechnisch zu nutzen, wird das web-basierte Datamining Tool OverPass Turbo (http://overpass-turbo.eu/) verwendet. Hierbei kann die OSM zugrundeliegende Geodatenbank nach spezifischen Informationen gefiltert werden. Diese können anschliessend in einem GeoJSON exportiert und weiterverwendet werden.

Repository

Der Code ist in den Jupyter Notebooks Schritt für Schritt dokumentiert. Die hierbei verwendete Sprache ist Englisch. **GitHub** Repository: https://github.com/LucienOST/PT1-Refuge_Islands

4 Modell Implementierung

In diesem Kapitel sind die, aus Sicht des Autors, wichtigsten Schritte, Konzepte und Ergebnisse des Praxisteils dokumentiert. Eine detailliertere Prozessanalyse befindet sich in den Jupyter Notebook Dateien, in welchen der Code erfasst ist. Wie zu Beginn dieser Arbeit erwähnt, wurde der Fokus dieser Dokumentation speziell auf die Bereiche gelegt, welche sich von der Vorarbeit unterscheiden, oder in der Vorarbeit wenig bis gar nicht dokumentiert wurden. Gleich wie im theoretischen Teil dieser Arbeit, ist es auch im praktischen Teil das Ziel, die sich mit der Vorarbeit überschneidenden Bereiche zu ergänzen.

4.1 Phase 1 – Vorbereitung

Ziel der ersten Phase ist es, das Datenset aufzubereiten. Dies beinhaltet die Beschaffung und Bereinigung der Daten aber auch die Erstellung von binären Masken, welche als Labels für das Modelltraining benötigt werden. Anders als in der Vorarbeit, sind die Labels diesmal im Datenset nicht enthalten. Ebenfalls Teil der Vorbereitungsphase ist die anschliessende Zuteilung der Daten in das Trainings- oder Testset.

4.1.1 Datenbeschaffung

Das Datenset wird vom Geometa Lab der OST als Zipdatei über Switch Drive für den Download bereitgestellt. Darin enthalten sind über 4000 Orthofotos im TIF Format und eine CSV Datei. Die Bilder sind anhand ihrer Zentralen Koordinaten in **WSG84 Web Mercator** Form benannt. Die Tabelle der CSV Datei enthält die folgenden Spalten:

- Link zu den jeweiligen TIF Bilder in Form eines **Dateipfades**
- **Polygon** Koordinaten des betroffenen Bildes
- **Invalid_Image**: Boolean Wert bezüglich der Gültigkeit des Bildes (True / False)

Um die Daten im späteren Verlauf effizient nutzen zu können, muss in einem ersten Schritt ein strukturiertes Arbeitsverzeichnis erstellt werden. Anschliessend kann die Zip-Datei entpackt werden und die Bereinigungsphase beginnt.

4.1.2 Bereinigung und Formatierung

Die CSV Datei enthält auch **ungültige Einträge**. Diese zeigen sich in verschiedener Form. Im einfachsten Fall sind die betroffenen Zeilen mit einem entsprechenden Wert in der Invalid_Image Spalte gekennzeichnet. Teilweise haben aber auch als gültig gekennzeichnete Einträge keine Polygon Koordinaten oder die referenzierte Datei ist als Bild im Datenset gar **nicht enthalten**. Letzteres Problem besteht auch in umgekehrter Form. Bild-Dateien haben keinen Eintrag im CSV-File. Diese ungültigen Einträge und Bilder müssen entfernt werden, da für sie keine Masken erstellt werden können und daher für das Modell **nicht verwendbar** sind.

Die enthaltenen TIF-Dateien sind gross und nicht weiter für den Bildverarbeitungsprozess geeignet. Daher wird jedes Orthofoto komprimiert, in ein **PNG** Format umgewandelt und separat abgespeichert. Bei über 4000 Bildern ist dieser Prozess entsprechend rechnerintensiv. Das Python Paket **Multiprocessing** biete hierfür aber Abhilfe. Dadurch wird der Konvertierungsprozess in mehrere Kindsprozesse unterteilt, welche parallel abgearbeitet werden. Dies ist jedoch nur, bei einen Computer mit Mehrkernprozessor (Pythontic 2022).

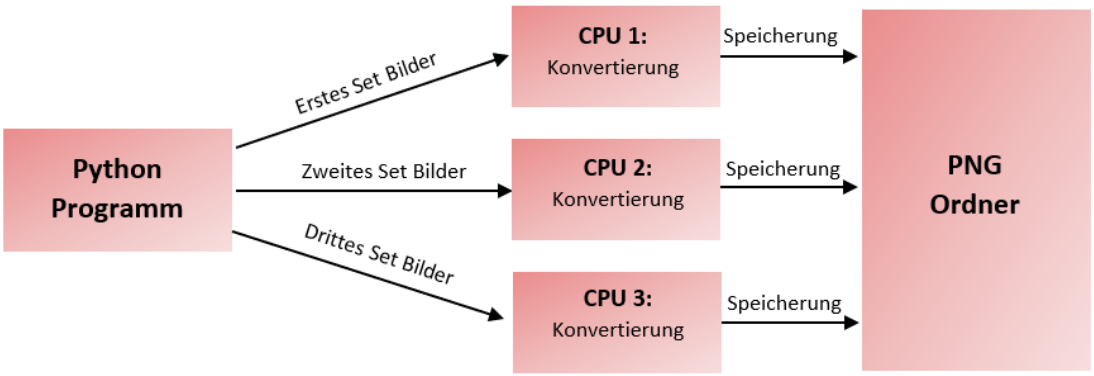


Abbildung 12: Konzept des Multiprocessing in Python - eigene Darstellung basierend auf (Pythontic 2022)

4.1.3 Maskenerstellung

Mit den bereinigten Daten können nun die **Labels** generiert werden, welche für das Trainieren des Modelles benötigt werden. Hierfür wird zuerst der in der CSV Datei enthaltene Dateipfad mit dem des referenzierten PNG Bildes abgeglichen. In einem zweiten Schritt wird ein neues Bild im äquivalenten Format des PNG Bildererstellt, wobei alle **Pixelwerte** vorerst den Wert 0 haben. Anhand der Polygon Koordinaten im CSV File können nun die **Pixelkoordinaten**, welche sich innerhalb des Polygons befinden auf den Wert 1 gesetzt werden. Das Resultat ist Binär-Bild das wiederum als PNG im Masken Ordner gespeichert wird.

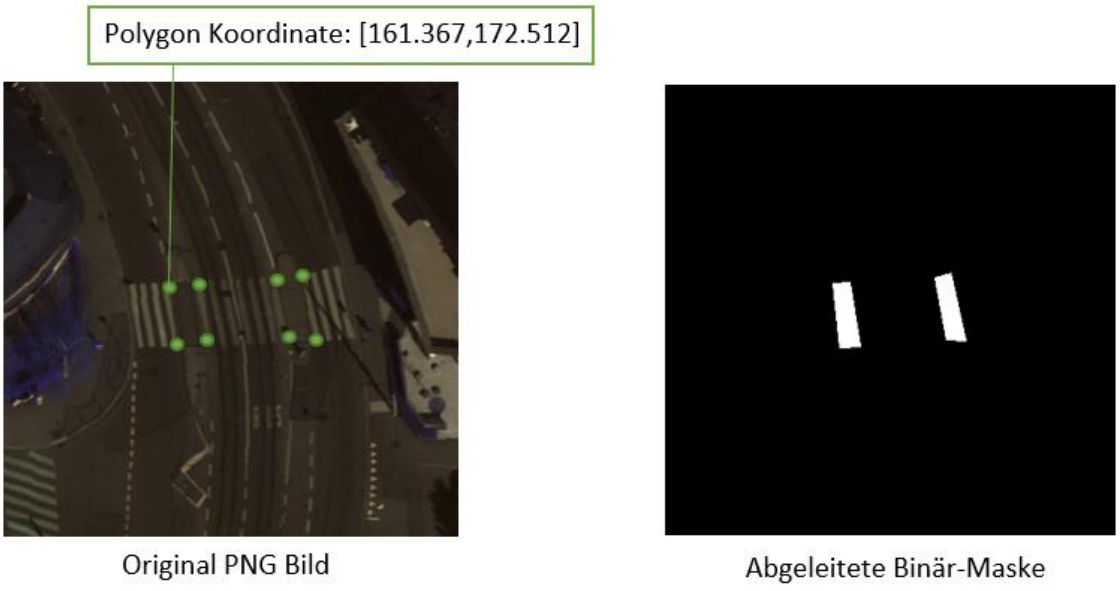


Abbildung 13: Polygon-Koordinaten und daraus erstelle Masken

4.1.4 Set Zuteilung

Als letzter Schritt der Vorbereitungsphase folgt nun die Zuteilung der PNG Bilder und Masken. Gemäss ML-Theorie sollten diese Daten in einem ungefähren Verhältnis von 80% für das Trainingsset und 20% für das Testset aufgeteilt werden (Helmut Grabner und Christoph Würsch 2021). Dieser Prozess wird grösstenteils von der Deep Learning Bibliothek Fastai übernommen. Nur die entsprechenden Verzeichnisse müssen vorab noch erstellt werden. Nach Beendigung dieses Schrittes sollten die Daten wie folgt strukturiert sein:

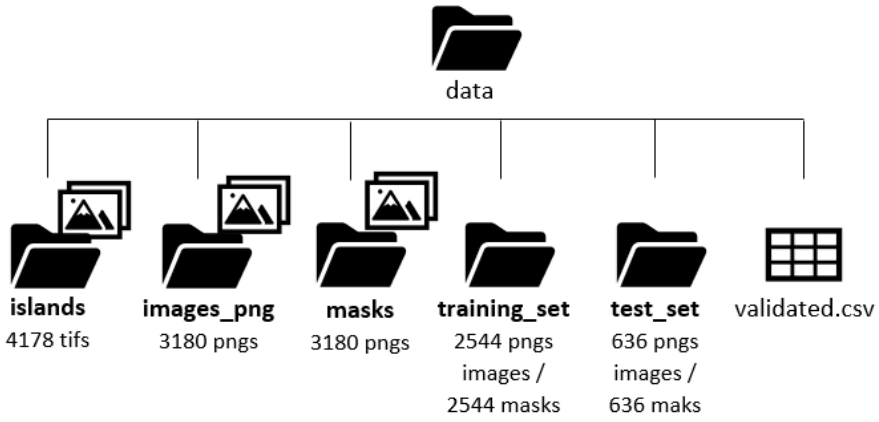


Abbildung 14: Übersicht über das Datenset nach Abschluss der Vorbereitungsphase

4.2 Phase 2 – Modellimplementierung

In Phase Zwei beginnt mit der Konfiguration des Convolutional Neural Network. Anschliessend wird dieses mit den Bildern und Masken der vorherigen Phase trainiert und getestet. Die Arbeitsschritte orientieren sich hierbei stark an den erlernten Vorgehensweisen des Fastai Online Kurses.

4.2.1 Modell Konfiguration

Durch die Verwendung der Fastai Bibliothek wird der eigentlich Modellbau stark abstrahiert und vereinfacht. Um ein Modell in Fastai zu initialisieren, muss jedoch zuerst festgelegt werden, wie der Output generiert und wie die vorhandenen Labels interpretiert werden sollen. Dies erfolgt durch sogenannte DataBlocks von Fastai. Anschliessend werden diese Zusammen mit den Trainingsbildern in ein Fastai Modell (in Fastai **Learner**) geladen. Dort kann vortrainiertes Convolutional Neural Network direkt über die Fastai API importiert werden. Somit wird auch bereits die Architektur des Modells festgelegt. Wie bereits in der Vorgängerarbeit wurde ein **ResNet18**, also ein Residuales Netzwerk mit 18 versteckten Schichten, verwendet. Das Importierte ResNet18 ist bereits darauf trainiert High-Level Merkmale wie Kanten und Ecken in Bildern zu erkennen. Für die eigentliche Aufgabe, Fussgängerinseln auf Luftaufnahmen zu segmentieren, wird lediglich eine neue Schicht am Ende des neuronalen Netzwerks hinzugefügt. Fastai liefert also das Grundgerüst. Trotz der Abstraktion müssen gewissen Konfigurationen selbst spezifiziert werden. Die zwei wichtigsten sind:

1. **Datenerweiterung**: Um das volle Potential der vorhanden Trainingsdaten auszuschöpfen, kann dem Modell erlaubt werden, leichte Variation der Input Bilder zu erstellen. Dies kann beispielsweise durch Rotation des Bildes Veränderung der Lichtverhältnisse erreicht werden. Durch diese Erweiterung des Datensatzes kann das Modell besser trainiert werden und letztlich bessere Ergebnisse liefern. Diese Option muss in Fastai speziell deklariert und anschliessend dem Modell zugewiesen werden.
2. **Performance Metrik**: Um eine Aussage über die Performance eines Modells treffen zu können, müssen bestimmte Berechnungen durchgeführt werden. Bei der semantischen Bildsegmentierung werden die Output-Masken (**predicted Masks**) mit den Ziel-Masken **(target Masks**) verglichen. Damit dieser Vergleich stattfinden kann müssen vorab beide Dateien normalisiert werden.

4.2.2 Modell Training und Nachjustierung

Sobald die Modelparameter festgelegt sind, kann mit dem Training begonnen werden. Mit allen Bildern gleichzeitig zu trainieren ist wie erwähnt nicht zweckmässig. Ein Trainingsdurchlauf würde zu lange dauern. Auch muss zuerst getestet werden ob der Prozess funktioniert, das heisst es muss geprüft werden, ob sich das Modell effektiv verbessert. Daher werden vorher kleiner Batches definiert, die zufällig mit Bildern aus dem Dataset gefüllt werden. Die Batch-Grösse kann hierbei variieren. Zu Beginn wurde mit kleinen Grössen gearbeitet, dann wurden sukzessive mehr Bilder hinzugefügt. Ebenfalls wurde die Anzahl Trainingsdurchläufe (Epochen) schrittweise erhöht. In *Abbildung 15* ist zu erkennen, wie sich die Leistungs-Metriken nach jeder Epoche verändern. Nach jeder Epoche werden die Gewichte der letzten Schicht angepasst. Die Funktion **freeze()** bzw. **unfreeze**() sorgt dafür, dass die restlichen, bereits vortrainierten Schichten des von Fastai verwendeten Neuronalen Netzes nicht verändert werden. Die Genauigkeit (Accuracy) wird zu Beginn schnell erhöht, während der Verlust (Loss) schnell sinkt. Nach den ersten Durchläufen verändern sich diese Werte nur noch in kleineren Schritten. Die Genauigkeit muss nicht mit jeder Epoche besser werden, doch die Tendenz sollte positiv sein. Wenn dies nicht der Fall ist, ist dies ein Indiz dafür, dass das Modell nicht korrekt lernt (underfitting) und die Modelparameter müssten entsprechend überarbeitet werden.

```
learn.fit_one_cycle(4,5e-3)
learn.unfreeze()
learn.fit_one_cycle(25,lr_max = slice(1e-6,1e-4))
```

epoch	train_loss	valid_loss	calc_accuracy	time
0	14.808592	0.019832	0.089812	02:13
1	0.015428	0.010720	0.516481	02:13
2	0.010131	0.008949	0.577732	02:16
3	0.007220	0.007035	0.728739	02:16

epoch	train_loss	valid_loss	calc_accuracy	time
0	0.007357	0.006987	0.740200	02:18
1	0.006406	0.006883	0.734939	02:19
2	0.007750	0.006877	0.779731	02:19
3	0.005983	0.006639	0.752450	02:19
4	0.006459	0.006419	0.760159	02:19
5	0.006173	0.006343	0.780071	02:20
6	0.006517	0.006055	0.775187	02:20
7	0.005605	0.006128	0.777016	02:18
8	0.006660	0.005798	0.792072	02:18

Abbildung 15: Epochen und Metriken beim Trainingsprozess

Das Training mit über 2500 verfügbaren Luftbildern war Zeitintensiv. Entsprechend zog sich der ganze Prozess über mehrere Wochen. Die berechnete Parametrisierung des Modells kann nach jedem Durchlauf abgespeichert werden, so dass kein Trainingsfortschritt verlorengeht. Zu einem später Zeitpunkt kann der Prozess fortgesetzt werden. Auch das ganze Modell kann exportiert werden, damit es an einem anderen Tag wiederverwendet werden kann. Ab wann ein Modell ausreichend trainiert ist, ist relative von der Aufgabenstellung und denn daran geknüpften Erwartungen. In der Theorie des Maschinellen Lernens gilt ein Modell als ausreichend Trainiert, wenn eine Genauigkeit von über 75% erreicht wurde (Helmut Grabner und Christoph Würsch 2021).

4.2.3 Modell Test und Vorhersagen

In der Testphase werden dem Neuronalen Netz Luftbilder übergeben, welche es noch nicht kennt. Somit kann dessen Eignung für die reale Anwendung evaluiert werden. Als Output erstellt das Modell nun selbst Masken (**Predicted Masks**) in der Form von Binär-Bilder. In diesen sind nur diejenigen Pixel auf den Wert 255 gesetzt welche das Modell als Teil einer Fussgängerinsel vermutet. Alle anderen Pixel sind auf 0 gesetzt. Nun kann ein direkter Vergleich mit den Ziel-Masken (**Target Masks**) durchgeführt werden. Insgesamt sind im Testset 636 Ziel-Masken, 654 Fussgängerinseln und über 170 Millionen Pixel, die zu Fussgängerinseln gehören, enthalten. Wichtig ist zu beachten, dass beide Masken im gleichen Format sind. Ansonsten stimmen die Pixelkoordinaten nicht überein und es kann keine Aussage über die Modellleistung getroffen werden. Die Ergebnisse können auf mehreren Ebenen analysiert werden:

		Absolute	Relative (%)
Mask	Total Masks	636	100.00
	Matched Masks (Accuracy)	580	91.19
Objects	Total Crossings (Ground Truth)	654	100.00
	Matched Crossings	606	92.66
	Precision		98.20
	Recall		92.70
	F1-Score		95.40
Pixel	Total Pixel	171720000	100.00
	Matched Pixel (Accuracy)	171265041	99.74
	Precision		78.55
	Recall		86.38
	F1-Score		80.68

Abbildung 16: Testergebnisse des Modells

Maskenebene: Insgesamt wurden von 636 vorhanden Masken 580 richtig prognostiziert, was einer Genauigkeit von **91.19%** entspricht. Dieser Wert kann als sehr gut bewertet werden.

Objektebene: In einigen Masken befinden sich auch mehrere Fussgängerinseln. Die totale Anzahl Fussgängerinseln auf den verwendeten Luftbildern beträgt 654. Davon wurden 606, also **92.66%**, vom Modell richtig erkannt.

Pixelebene: Die Pixeleben geht noch weiter ins Detail. Hier wird gemessen wie viele, der zu einer Fussgängerinsel gehörenden Pixel, richtig klassiert wurden -sprich den Pixelwert 255 erhalten haben. Für **99.74%** war dies der Fall. Eine Erklärung für dieses, im Vergleich zu den vorherigen Werten, bessere Ergebnis könnte sein, dass insbesondere kleine Fussgänger Inseln, mit entsprechen weniger Pixel, nicht richtig erkannt wurden.

Die zusätzlich berechneten Metriken **Precision**, **Recall** und **F1-Score** liefern eine vertieftere Analyse über die Perfomance des Modells. Diese Berechnung kann auf allen drei Ebenen der semantischen Segmentierung erfolgen (Masken, Objekte oder Pixel). Hierbei werden nicht nur die korrekt klassifizierten Bereiche einer Fussgängerinseln (True Positive – TP) berücksichtig. Auch die inkorrekt klassifizierten (False Positive – FP), sowie die fälschlicherweise **nicht-klassifizierten** Bereiche (False Negative – FN) sind relevant. Das bedeutet, dass die Fehler des Modells auch gewichtet werden. Zusammengefasst die einzelnen Berechnungsformeln und ihre Interpretation:

Precision: $PC = \frac{\mathrm{TP}}{\mathrm{TP+FP}} = \frac{\text{korrekt als Fussgängerinsel klassifizierter Bereich}}{\text{korrekt und inkorrekt als Fussgängerinsel klassifizierter Bereich}}$

Eine hohe Modellgenauigkeit kann irreführend sein. Wenn beispielsweise alle Pixel als Fussgängerinsel klassifiziert werden, sind konsequenterweise alle Fussgängerinseln erkannt. Eine Aussage darüber, ob das Modell nun die gewünschten Objekte von anderen Objekten zu unterscheiden vermag, kann jedoch nicht getroffen werden. Daher müssen die korrekt klassifizierten Bereiche in Relation zu allen als Fussgängerinseln klassifizierten Bereiche gesetzt werden.

Recall: $RC = \frac{\mathrm{TP}}{\mathrm{TP+FN}} = \frac{\text{korrekt als Fussgängerinsel klassifizierter Bereich}}{\text{korrekt klassifizierter und inkorrekt nicht−klassifizierter Bereich als Fussgängerinsel}}$

Die Recall Metrik fokussiert sich hingegen nur auf die Bereiche welche effektiv zur gesuchten Klasse gehören. Es ist ein Indiz dafür, wie genau das Modell die relevanten Objekte erkennt. Die vorhergesagten True-Positive Werte, wird ins Verhältnis zur effektiven Anzahl gesetzt.

F1-Score: $F1 = \frac{2*\mathrm{PC}*\mathrm{RC}}{\mathrm{PC+RC}}$

Der F1-Score kombiniert die Werte für Precision und Recall. Durch das Harmonisieren des Mittelwertes wird dafür gesorgt, dass keine der beiden Metriken die andere zu stark beeinflusst. Der F1-Score ist insbesondere für den Vergleich von zwei Modellen geeignet.

Wie in *Abbildung 15* zu erkennen ist, weist das Modell in allen gemessen Metriken ein gutes Ergebnis auf und kann als einsatzbereit erachtet werden. Der Praxisteil, welcher sich auf den Bereich des Maschinellen Lernens bezieht, endet hier. Nun erfolgt die Nachbearbeitung mit den vom Modell vorhergesagten Fussgängerinseln.

4.3 Phase 3 – Nachbearbeitung

Die Nachbearbeitungsphase besteht aus zwei Schritten. In Schritt Eins werden zuerst die geographischen Koordinaten der vom Modell prognostizierten Fussgängerinseln extrahiert und in einer GeoJSON Datei abgespeichert. In Schritt Zwei wird OpenStreetMap mit diesen Koordinaten abgeglichen und auf Unvollständigkeit geprüft.

4.3.1 Koordinaten Extraktion

Die vom Modell vorhergesagten Mittelinseln von Fussgängerstreifen sind als Binär-Bilder im PNG Format abgespeichert. Auf diesen ist in erste Linie nur ersichtlich wo im Bild sich das Objekt befinden. Geographische Koordinaten lassen sich aus dieser Dateiform jedoch noch nicht ableiten. Diese Information kann nur aus den geocodierten TIF-Orthofotos extrahiert werden. Für diesen Zweck müssen zuerst beide Dateien, die PNG Maske und das äquivalente TIF-Bild, in das gleiche Format transformiert werden. Dies hat zum Zweck, dass diejenigen Bildpunkte, die im PNG Bild die Fussgängerinsel darstellen, an der gleichen Pixelposition sind, wie in der TIF-Datei. Ohne diese Angleichung, wären die Koordinaten fehlerhaft. In dem geocodierten TIF-Bild referenziert jede Pixelkoordinate auch einen entsprechenden geographischen Punkt auf der Erde. Jedoch setzten sich die Fussgängerinsel in den PNG Masken aus zahlreichen Bildpunkten zusammen. Aus diesem Grund müssen ebenfalls noch die Zentroiden der Mittelinseln berechnet werden. So kann sichergestellt werden, dass nur ein Punkt, bzw. eine Koordinate, extrahiert wird. Im Gegensatz zum Datensatz der Vorarbeit, verwenden die hier vorhanden Orthofotos nicht das LV95 Format, sondern das Koordinatenreferenzsystem EPSG:3857. Hierbei handelt es sich ebenfalls um WGS84 Koordinaten, jedoch im Web Mercator Format. Dieses wird zum Rendern von Karten in Google Maps, OpenStreetMap und anderen Kartentools verwendet. Der wesentliche Unterschied: Die Längen- und Breitengrade des geographischen WGS84 System basieren auf einem ellipsenförmigen Ansatz, das Web Mercator WGS84 System hingegen auf einem spiralförmigen. Das Format der Koordinaten ist ähnlich, doch der Wertebereich ist unterschiedlich (Leah Wasser 2019).

Um sicherzustellen, dass die berechneten Koordinaten korrekt sind, werden mehrere, zufällige Stichproben auf Google Maps und OpenStreetMap durchgeführt. Alle erwiesen sich als korrekte Koordinaten einer Fußgänger Insel. Jedoch gilt folgendes zu beachten:

- Der berechnete Koordinatenpunkt ist nicht in jedem Fall der exakt zentroide Punkt der Mittelinsel. Diese Abweichung muss für den spätern Vergleich berücksichtigt werden, da es zwangsläufig kleiner Unterschiede bei den effektiven Werten für Längen- und Breitengrade geben wird. Diese enthalten sechs Dezimalstellen und sind metergenau.

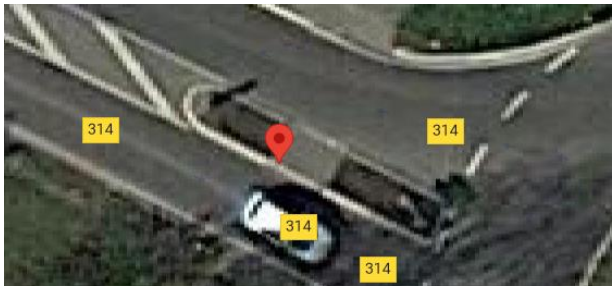


Abbildung 17: Lokalisierung einer vom Modell prognostizierten Punktkoordinate

- Bei der Verifizierung, ob es sich bei der Vorhersage effektiv um eine Fussgänger-Insel handelt, zeigte sich, dass die Tools nicht immer übereinstimmen, bzw. gewisse Tools weniger aktuell sind. Eine Vorhersage wurde daher immer in beiden Tools überprüft.

Google Maps Satellitenaufnahme /
WGS84 Koordinaten: [47.178285, 9.522092]

OSM Satellitenaufnahme /
WGS84 Koordinaten: [47.178285, 9.522092]

Abbildung 18: Unterschiedliche Aktualität bei Google Maps und OpenStreetMap

Auf dem Satellitenbild von Google liess sich zuerst keine Fussgäner-Insel erkennen. Dies kann zu einem Fehlschluss führen. Auf OSM zeigt sich hingegen, dass die Vorhersage des Modells für diese Koordinaten korrekt war. Tatsächlich wurden in diesem Bereich gleich mehrere Mittelinseln erkannt. Auf OSM waren die Objekte auch bereits entsprechend getaggt:

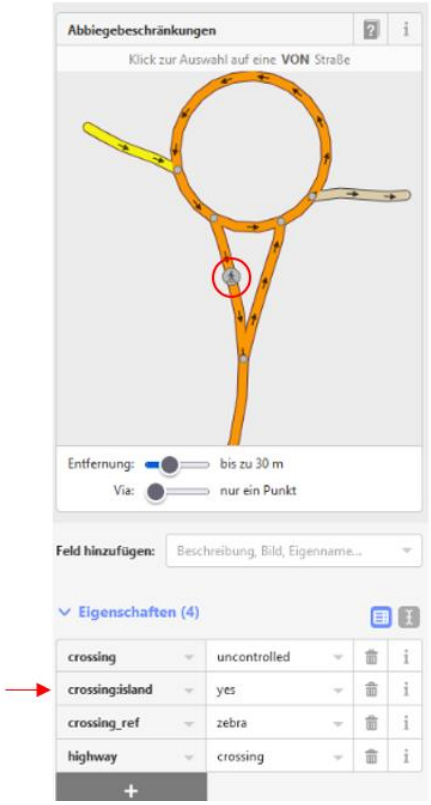


Abbildung 19: Korrekte Markierung einer Fussgänger Insel in OpenStreetMap (OSM Org. 2022)

4.3.2 OpenStreetMap Abgleich

Bei allen Stichproben haben sich die Prognosen des Modells und die extrahierten Koordinaten als richtig erwiesen. Daher können diese für einen Abgleich mit OSM als geeignet erachtet werden. Hierfür müssen zuerst die Koordinaten der in OSM bereits erfassten Fussgänger-Inseln geladen werden. Dies erfolgt über die Overpass Turbo API welche direkt in Python verwendet werden kann. Mit einer entsprechenden API-Abfrage kann die Geodatenbank von OSM nach spezifischen Einträgen gefiltert werden. Die Ergebnisse lassen sich anschliessend direkt als GeoJSON Datei speichern.

Für den Abgleich der Prognosen werden zwei Region näher untersucht. Zum einen die Ostschweiz, das heisst die Kantone St.Gallen, Appenzell Inner- und Ausserrhoden, Thurgau, Schaffhausen, Glarus und Graubünden in einer kombinierten Abfrage. Zum anderen der Kanton Zürich allein. Um festzustellen, ob wirklich alle vom Modell vorhergesagten Mittelinseln in OSM erfasst sind, werden zum Schluss Einträge in der ganzen Schweiz geladen. Kantone können beliebig kombiniert werden.

```
api = overpass.API()
result = api.get("""(area["name"="Sankt Gallen"];
                     area["name"="Thurgau"];
                     area["name"="Appenzell Innerrhoden"];
                     area["name"="Appenzell Ausserrhoden"];
                     area["name"="Schaffhausen"];
                     area["name"="Glarus"];
                     area["name"~"Graubünden"]
                     );
                     (nw["crossing:island"="yes"](area););""")
with open('CH_East_Islands.geojson', 'w') as fd:
    geojson.dump(result,fd)
print("File successfully stored under: " + str(cwd))
```

Abbildung 20: Overpass Turbo Abfrage in Python

Die GeoJSON Dateien müssen zuerst entpackt werden damit die Koordinaten aus dem File weiterverwendet werden können. Anschliessend werden diese in ein Numpy Array geladen und für den Abgleich aufbereitet. Wie im vorherigen Abschnitt erwähnt, muss beachtet werden, dass die zu vergleichende Werte nicht bis zur letzten Dezimalstelle übereinstimmen. Die Abweichen lassen sich vermutlich auf kleinere Ungenauigkeiten des Modells zurückführen. Wie in der Testphase ersichtlich, prognostiziert dieses nicht 100% korrekt. Ob in OSM die Inseln jeweils immer an ihrem zentrierten Punkt getaggt sind, ist nicht ersichtlich. Die Koordinaten werden daher mit einer Toleranz von zuerst 1e-5 verglichen. Danach wird die Toleranz sukzessive erhöht, um grössere Abweichungen festzustellen. Der Vergleich hat folgende Ergebnisse geliefert:

		OpenStreeMap		
	Modellprognosen	Ostschweiz	Zürich	Ganze Schweiz
Enthaltene Inseln	659	1161	2326	9089
Prognostizierte Inseln vom Modell		158	425	623

Tabelle 1: Vergleich der prognostizierten Objekte mit OpenStreetMap

- 158 Mittelinseln befinden sich nicht in der Ostschweiz oder dem Kanton Zürich
- 36 Mittelinseln, die vom Modell vorausgesagt wurden, konnten keinem Eintrag in OSM für die Region Schweiz zugewiesen werden.

Diese 36 Koordinaten werden nun manuell überprüft. Zu diesem Zweck werde diese wieder in ein separates GeoJSON exportiert. Mithilfe von GeoJson.io (https://geojson.io/) können die Koordinaten nun auf einer Karte genauer überprüft werden. Es stellt sich heraus, dass sich alle nicht zugewiesenen Mittelinseln im nahen Grenzgebiet, ausserhalb der Schweiz befinden. Konsequenterweise wurde diese Punkt bei der Overpass API-Abfrage herausgefiltert. Einzelne Stichproben auf OSM haben aber gezeigt, dass auch diese Objekte bereits korrekt erfasst und entsprechend als Fussgänger-Insel markiert sind.

Abbildung 21: Lokalisierung nicht übereinstimmender Koordinaten

5 Ergebnisse

Der erarbeitete Ansatz von Kevin Ammann wurde durch den Prozess der Maskengenerierung und den Abgleich mit OSM ergänzt. Diese Arbeit zeigt, dass das Verfahren für verschiedene Objektklassen funktioniert und mit kleinen Schritten ausgebaut werden kann. Das Gesamtergebnis dieser Arbeit ist ein computergestützter, fast vollständig automatisierter Prozess zur Objektsegmentierung in Luftbildern und zur anschliessenden Unterstützung bei Überprüfungsaufgaben von Kartentools. Durch die korrekte Aufbereitung der Orthofotos, sowie durch eine passende Konfiguration und gezieltem Training des künstlichen neuronalen Netzes, ist das verwendete Modell nun fähig, Fussgängerinseln zuverlässig zu erkennen. Zudem konnten weitere Information aus den Luftbildern extrahiert werden. Der gesamte Datenfluss über die drei Projektphasen lässt sich anhand von folgendem Diagramm ableiten:

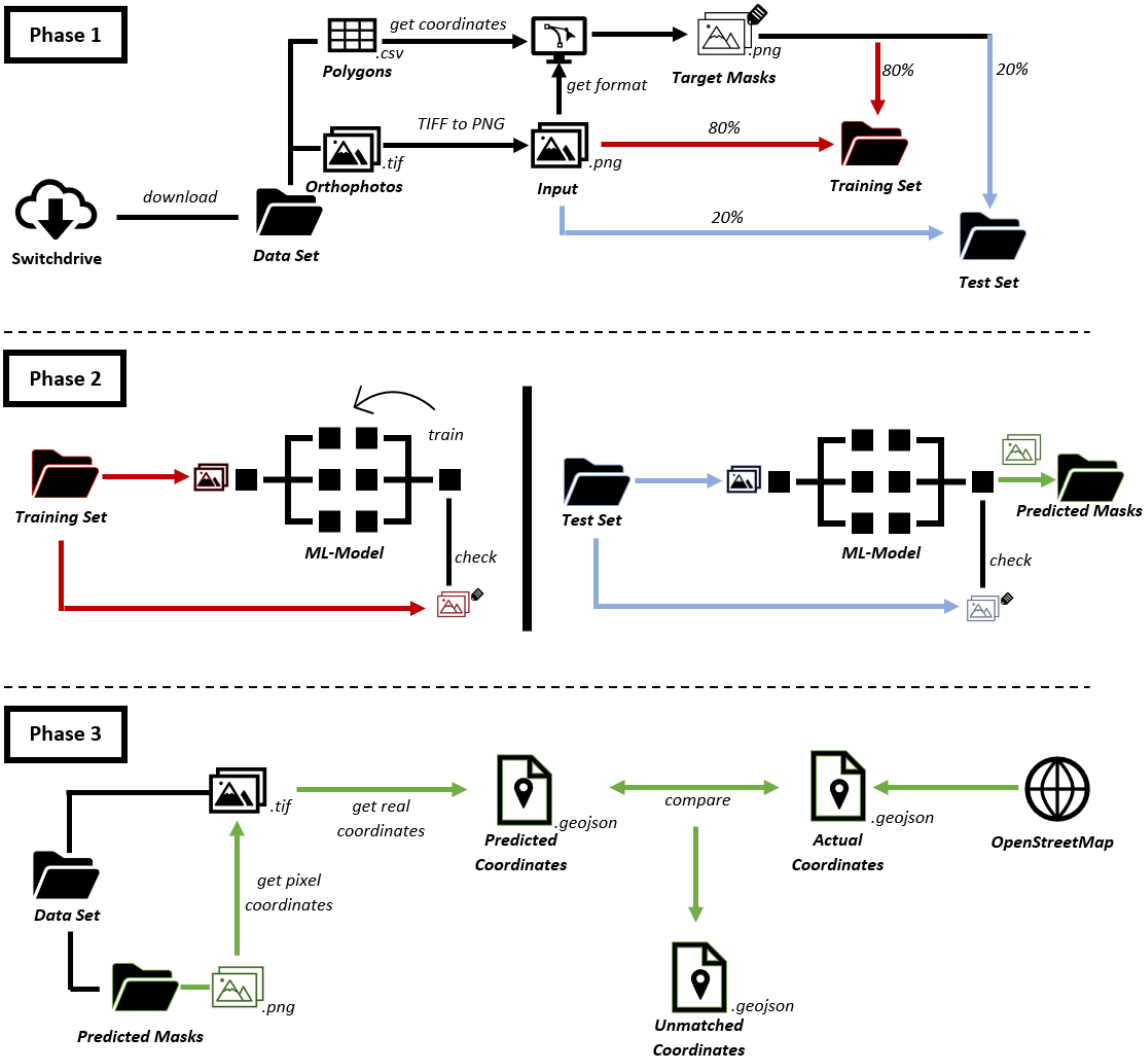


Abbildung 22: Datenfluss Diagramm über drei Prozessphasen

Die implementierten Prozessschritte der Nachbearbeitung ermöglichen es, diese Information direkt für eine erste Analyse weiterzuverwenden. Ein erster Abgleich mit OSM für verschiedene Regionen der Schweiz wurde durchgeführt. Für die erhaltenen Orthofotos kann festgestellt werden, dass alle erkannten Mittelinseln für Fussgängerstreifen -welche sich in der Schweiz befinden- in OSM bereits korrekt erfasst sind.

5.1 Zielüberprüfung

Zur Zielüberprüfung dieses Projektes, wird jedes gesetzte Ziel der Anfangsphase mit den erarbeiteten Ergebnissen und gewonnenen Erkenntnissen abgeglichen.

1. **Ziel:** Aneignung der Grundkonzepte des Maschinellen-Lernens, insbesondere in den Bereichen Computer Vision, mit Fokus auf Bildsegmentierung mittels Deep Learning und Convolutional Neural Networks.

1.1. **Überprüfung:** Die Disziplin des Maschinellen-Lernens ist sehr umfassend. Entsprechend konnten nicht alle Grundkonzepte in dieser Arbeit behandelt werden. Jene, die für diese Arbeit relevant sind, wurden jedoch erlernt und sind kurz, aber dennoch nachvollziehbar dokumentiert. Der behandelten Themen ergänzen auch gut jene der Vorarbeit.

2. **Ziel:** Erlernen nützlicher Grundlagen im Bereich Geoinformationssysteme und Geodaten.

2.1. **Überprüfung:** Die Vorkenntnisse des Autors in diesem Bereich waren gering. Daher konnte dieses Thema nicht sehr vertieft behandelt werden. Dieser Theorieteil ist im Vergleich zur ML-Domäne kürzer ausgefallen. Nur die wichtigsten Grundlagen und Verfahren, die für das Verständnis und die Umsetzung des Praxisteils benötigt werden, wurden behandelt. Diese sind aber verstanden und konnten korrekt umgesetzt werden.

3. **Ziel:** Anpassung des bestehenden Modells sowie anschliessendes Training mit Bildern von Fussgängerinseln.

3.1. **Überprüfung:** Der Prozess der Vorgängerarbeit wurde erlernt und konnte dadurch zweckmässig angepasst werden. Das Modell wurde mit anderen Bildern trainiert, bis es einsatzfähig war.

4. **Ziel:** Testen des Modells und Auswertung der Ergebnisse.

4.1. **Überprüfung:** Das Modell hat in der Testphase gute Werte erzielt. Ebenfalls konnten die erstellten Prognosen für weitere Analysen verwendet werden.

5. **Ziel:** Anhand der erzielten Ergebnisse einen Ansatz zum Abgleich mit OSM-Daten erstellen.

5.1. **Überprüfung:** In der Nachbearbeitung konnte ein Ansatz implementiert werden, um die extrahierte Information direkt mit OSM abzugleichen.

6. **Ziel:** Dokumentation des gesamten Implementierungsprozesses und des Datenflusses. Vom Erhalt des Datensatzes bis zum Abgleich mit OSM.

6.1. **Überprüfung:** Der Prozess ist schrittweise dokumentiert. Ein Datenflussdiagramm wurde erstellt und der Code in den Jupyter Notebook Dateien wurde mit Anmerkungen ergänzt.

5.2 Verbesserungen

Modelverwendung: Anstatt direkt das ResNet18 von Fastai auf Fussgänger Inseln zu trainieren, hätte das bereits trainierte neuronale Netz von Kevin Ammann ausgebaut werden können. Dieses ist bereits für die Erkennung von Fussgängerstreifen geeignet und hätte mit einer neuen Neuronen Schicht nochmals spezifischer trainiert werden können. Da es zu Beginn jedoch Probleme beim Laden der Learner-Datei gab, wurde auf das vorhandene ResNet18 Modell von Fastai zurückgegriffen. Da das Training zeitintensiv ist und auch eine entsprechende IT-Infrastruktur voraussetzt, reichte die Zeit am Ende nicht mehr, um zwei neuronale Netze zu trainieren und den ursprünglichen Plan doch noch umzusetzen. Planungsmässig hätte der zweiten Phase dieses Projekts mehr Zeit eingeräumt werden müssen.

Datenset: Viele Bilder des hier verwendeten Datensatzes konnte nicht für das Training und den Test des Modells verwendet werden. Entweder sind die Fussgänger Inseln nicht eingezeichnet (keine Polygon-Koordinaten) oder im Luftbild sind gar keine solche Objekte vorhanden sind. Infolgedessen erhielt das Modell weniger Bilder als ursprünglich vermutet und auch die Prognosen durch den Modell-Test sind weniger zahlreich. Mit mehr Orthofotos von Mittelinseln könnte ein umfassenderer Abgleich mit OSM erfolgen, welcher eine präzisere Aussage über die Aktualität des Tools ermöglichen würde.

5.3 Fortführung

Prinzipiell kann der hier vorliegende Ansatz für weitere Objektklassifizierungsaufgaben ausgeweitet werden. Da anhand dieser Arbeit jedoch nun erwiesen ist, dass das Vorgehen reproduziert und für unterschiedliche Objekte angewendet werden kann, besteht grösseres Potential in der Ausweitung der Nachbearbeitungsphase. Der vorliegende Abgleich mit OSM genügt für eine erste grobe Einschätzung bezüglich der Aktualität der Daten. Interessant wäre jedoch eine automatische Korrektur oder Überprüfungsmeldung direkt im Tool. Sprich ein Verfahren welches automatisch die aktuellen Luftbilder auf bestimmte Objekte überprüft und die Erkenntnisse und Einschätzungen direkt in OSM integriert. Dies wäre eine nützliche Unterstützung für die OSM-Community in ihrem Bestreben die digitale Weltkarte aktuell zu halten.

6 Reflexion

Die Projektarbeit war eine Herausforderung. Viele Konzepte mussten von Grund auf erlernt und direkt umgesetzt werden. Durch das Orientieren an einem bereits erarbeiteten Prozess, war die Freiheit der Umsetzung in gewisser Weise eingeschränkt. Das Wiederverwenden von nicht selbstgeschriebenem Programmiercode birgt ebenfalls Herausforderungen. Besonders, wenn der Code mit nur wenigen Anmerkungen versehen ist.

6.1 Erlerntes

Das direkte Anwenden der ML-Theorie in verschiedenen Projektphasen hat viel zum Verständnis dieser Disziplin beigetragen. Die recherchierten Konzepte mussten direkt anhand einer konkreten Aufgabe umgesetzt werden. Dies hat den Lerneffekt verstärkte. Ebenfalls zeigte sich dadurch, wie die Prozessschritte miteinander verknüpft sind und wie wichtig ein exaktes Arbeiten von Beginn an ist. Beim Erlernen der ML-Disziplin, sei dies den Theoriemodulen im Studium oder bei Online-Tutorials, wird hauptsächlich auf den Modellbau und das anschliessende Training und Testen fokussiert. Oft sind sauber aufbereite Datensätze bereits vorhanden. In der Praxis kann in vielen Fällen jedoch nicht auf ein solches zurückgegriffen werden, da für den spezifischen Anwendungsfall keines existiert. Korrekte Daten sind entscheidend für den Ausgang eines solchen Unterfangens. Daher ist es wichtig, sich auch mit der Vorbereitungsphase zu befassen. Ebenfalls gering thematisiert in der Theorie, ist die eigentliche Nutzung der vom Modell erhaltenen Vorhersagen. Meistens endet der Prozess nach einem zufriedenstellenden Testresultat. Wie die Fähigkeiten des Modelles nun konkret eingesetzt werden können, wird nicht behandelt. Durch die Aufgabenstellung für diese Arbeit war dies nicht der Fall. Vorbereitung und Nachbearbeitung standen ebenso im Fokus. Dies hat zu einem umfassenderen Verständnis geführt.

6.2 Kritik

Der Zeitaufwand für die einzelnen Phasen wurde teilweise unterschätzt. Auch bauen die meisten Arbeitsschritte aufeinander auf, so dass erst nach korrekter Beendigung einer Aufgabe mit der nächsten begonnen werden kann. Auch besteht bei einem ML-Projekt grosse Abhängigkeit vom Datenset. Für einen erfolgreichen Ausgang innerhalb eines Zeitplans müssen die Daten rechtzeitig verfügbar sein. Dies erfolgte bei diesem Projekt etwas verspätet. Zu Beginn des Projektes konnte die Wartezeit jedoch mit der Ausarbeitung des Theorieteils überbrückt werden.

Die Analyse von tausenden Bildern ist ein sehr leistungs- und zeitintensiver Prozess. Damit dies auf effiziente Weise durchgeführt werden kann, wird eine gewisse IT-Infrastruktur benötigt. Das Geometa Lab der OST konnte zwar einen Fernzugriff auf einen GPU-Cluster bereitstellen, jedoch hätte dies früher vom Autor dieser Arbeit beantragt werden müssen. Leider ging auch hier Zeit verloren.

Der Zeitplan konnte nicht ganz eingehalten werden. Daher, muss in Zukunft die Planung ausgereifter ausfallen, um einen fliessenden Arbeitsprozess zu ermöglichen.

7 Dank

Das Projekt wäre ohne die Unterstützung des Geomata Lab nicht möglich gewesen. Speziellen Dank geht daher an Prof. Stefan Keller, welcher diese Arbeit betreute und mir eine selbständige Arbeitsweise mit vielen Freiheiten ermöglichte. Ebenfalls bedanken möchte ich mich bei Nicola Jordan, welche bei Fragen und Unklarheiten stets Unterstützung bot, sowie bei den weiteren Teammitglieder, welche das Datenset erstellt und die tausenden Fussgänger Inseln mit Polygon-Koordinaten markiert haben.

8 Verzeichnisse

8.1 Literaturverzeichnis

Alessandro Giusti (2021): Introduction to Image Classification. SUPSI / ZHAW. Master of Science in Engineering. ZHAW Online, 19.10.2021, zuletzt geprüft am 04.07.2022.

Alexander Amini (2020): Convolutional Neural Networks. MIT, 2020. Online verfügbar unter https://www.youtube.com/watch?v=iaSUYvmCekI&t=1866s.

Artem Oppermann (2021): Regularisierung in Deep Learning: L1, L2 und Dropout. Online verfügbar unter https://artemoppermann.com/de/regularisierung-in-deep-learning-l1-l2-und-dropout/.

Bradski, Gary R.; Kaehler, Adrian (2011): Learning OpenCV. Computer vision with the OpenCV library. 1. ed., [Nachdr.]. Beijing: O'Reilly (Software that sees).

Daniel Trabold (2021): Welche Arten von Maschinellem Lernen gibt es? Hg. v. ML2R. Fraunhofer Strategischen Forschungsfeldes KI. Online verfügbar unter https://machinelearning-blog.de/grundlagen/welche-arten-von-maschinellem-lernen-gibt-es/.

Data Robot (2018): Introduction to computer vision: what it is and how it works. Hg. v. Data Robot. Online verfügbar unter https://www.datarobot.com/blog/introduction-to-computer-vision-what-it-is-and-how-it-works/.

datei.wiki:.tif Dateierweiterung. Online verfügbar unter https://datei.wiki/extension/tif.

fast.ai (2022): About fastai. Online verfügbar unter https://docs.fast.ai/.

geojson.org: GeoJSON. Online verfügbar unter https://geojson.org/.

Grant Sanderson (2020): Convolutions in image processing. 18.S191. MIT, 2020. Online verfügbar unter https://www.youtube.com/watch?v=8rrHTtUzyZA&t=147s.

Hans-Peter Bähr (Hg.) (2005): Digitale Bildverarbeitung. Anwendungen in Photogrammetrie, Fernerkundung und GIS. 4., völlig neu bearb. Aufl. Heidelberg: Wichmann.

Helmut Grabner; Christoph Würsch (2021): Machine Learning. MSE FTP Module. Master of Science in Engineering. ZHAW. Zurich, 2021.

Hmrishav (2022): An Introduction to Image Segmentation. Deep Learning vs. Traditional. Hg. v. V7. Online verfügbar unter https://www.v7labs.com/blog/image-segmentation-guide.

Ilija Mihajlovic (2019): Everything You Ever Wanted To Know About Computer Vision. Hg. v. Towards Data Science. Online verfügbar unter https://towardsdatascience.com/everything-you-ever-wanted-to-know-about-computer-vision-heres-a-look-why-it-s-so-awesome-e8a58dfb641e.

Inis Ehrlich (2022): Was ist ein neuronales Netz? Begriffe rund um die KI, Maschinelles Lernen und neuronale Netze erklärt. Hg. v. WFB Bremen. Bremer KI-Transfer-Zentrum. Online verfügbar unter https://www.wfb-bremen.de/de/page/stories/digitalisierung-industrie40/was-ist-ein-neuronales-netz.

Jared Lander; Michael Beigelmacher (2020): The Essential Guide to Quality Training Data for Machine Learning. What You Need to Know About Data Quality and Training the Machine. Hg. v. Cloudfactory. Online verfügbar unter https://go.cloudfactory.com/hubfs/02-Contents/2-eBooks/The-Essential-Guide-to-Quality-Training-Data-for-Machine-Learning.pdf.

Jens Kürbig; Martina Sauter (2006): Bildsegmentierung und Computer Vision. Seminar. Universität Ulm. Online verfügbar unter https://www.mathematik.uni-ulm.de/stochastik/lehre/ws05_06/seminar/ausarbeitung_sauter.pdf.

Joseph Nelson (2020): You Might Be Resizing Your Images Incorrectly. Hg. v. Roboflow. Online verfügbar unter https://blog.roboflow.com/you-might-be-resizing-your-images-incorrectly/.

Keiron O'Shea; Ryan Nash (2015): An Introduction to Convolutional Neural Networks. Aberystwyth University, Ceredigion,. Online verfügbar unter https://arxiv.org/pdf/1511.08458.pdf.

Kevin Ammann (2022): Segmentierung in hochaufgelösten Luftbildern am Beispiel von Fussgängerstreifen. Project Thesis. OST, Rapperswil. Institute for Software.

Kevin P. Murphy (2012): Machine Learning. A Probabilistic Perspective.

Leah Wasser (2019): Lesson 3. Coordinate Reference System and Spatial Projection. Intro to Coordinate Reference Systems. Hg. v. Earth Datascience. Online verfügbar unter https://www.earthdatascience.org/courses/earth-analytics/spatial-data-r/intro-to-coordinate-reference-systems/.

Matt Przybyla (2020): Would You Rather Be an NLP or Computer Vision Data Scientist? A closer look into these popular Data Scientist roles. Hg. v. Towards Data Science. Online verfügbar unter https://towardsdatascience.com/would-you-rather-be-an-nlp-or-computer-vision-data-scientist-5b1d45e1c601.

Michael Herter (Hg.) (2008): Handbuch Geomarketing. Heidelberg: Wichmann. Online verfügbar unter http://bvbr.bib-bvb.de:8991/F?func=service&doc_library=BVB01&doc_number=015676917&line_number=0001&func_code=DB_RECORDS&service_type=MEDIA.

OSM Org. (2022): About OpenStreetMap - OpenStreetMap Wiki. Online verfügbar unter https://wiki.openstreetmap.org/wiki/About_OpenStreetMap.

Pythontic (2022): Multiprocessing in Python. Online verfügbar unter https://pythontic.com/multiprocessing/multiprocessing/introduction.

R. Sablatnig; S. Zambanini, R. Licandro (2021): Einführung in Computer Vision. Hg. v. Technische Universität Wien. Online verfügbar unter https://www.cg.tuwien.ac.at/courses/EinfVisComp/Skriptum/SS14/EVC-02%20Einf%C3%BChrung%20in%20Computer%20Vision.pdf.

Rachel Zheng (2020): Image Segmentation: Predicting Image Mask with Carvana Data. Hg. v. Towards Data Science. Online verfügbar unter https://towardsdatascience.com/image-segmentation-predicting-image-mask-with-carvana-data-32829ca826a0.

Raghul Asokan (2021): Basics of Computer Vision. Interpolation and Resizing. Hg. v. Medium. Online verfügbar unter https://raghul-719.medium.com/basics-of-computer-vision-1-image-resizing-97fca504cd63.

Rohit Singh (2019): Where Deep Learning Meets GIS. Hg. v. ArcWatch. Online verfügbar unter https://www.esri.com/about/newsroom/arcwatch/where-deep-learning-meets-gis/.

Swisstopo (a) (2020): Geodaten für alle. Hg. v. Bundesamt für Landestopografie swisstopo. Schweizerische Eidgenossenschaft. Online verfügbar unter https://www.swisstopo.admin.ch/content/swisstopo-internet/de/topics/geoinformation/_jcr_content/contentPar/tabs/items/258_1588255061068/tabPar/downloadlist/downloadItems/259_1588255134346.download/Broschuere_Geodaten_fuer_alle_A5_D_2020_www.pdf.

Swisstopo (b) (- Kein Datum): SWISSIMAGE RS. Hg. v. Swisstopo. Online verfügbar unter https://www.swisstopo.admin.ch/de/geodata/images/ortho/swissimage-rs.html.

Thomas Koller (2021): Convnets for semantic segmentation. Semantic Segmentation: Metrics, Conv Nets. Computer Vision. Master of Science in Engineering. Zürich, 17.11.2021.

Thomas Paschke (2020): GIS & KI: Mit GeoAI Objekte in Bildern intelligent erkennen. Hg. v. Where Next. Online verfügbar unter https://wherenext.esri.de/gis-kunstliche-intelligenz-geoai/.

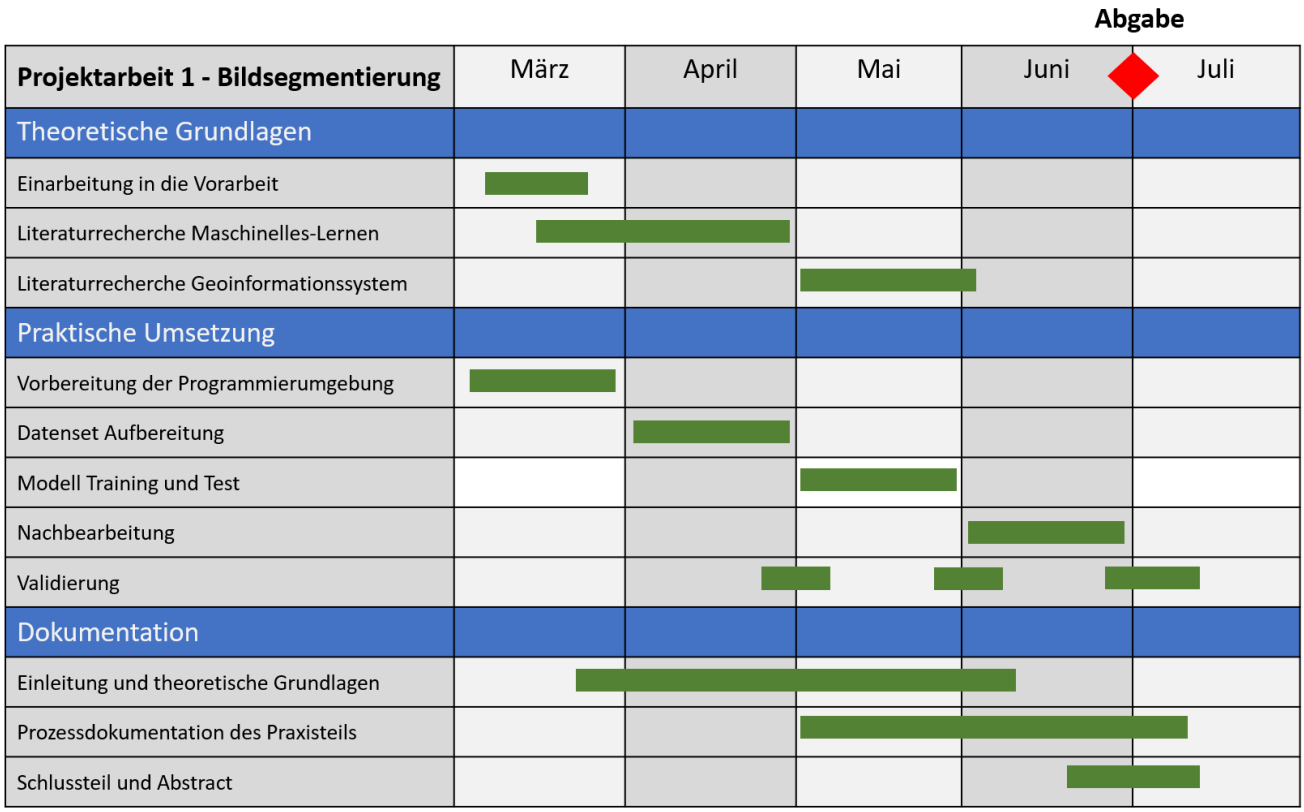
Valentina Alto (2019): Neural Networks: parameters, hyperparameters and optimization strategies. Hg. v. Towards Data Science. Online verfügbar unter https://towardsdatascience.com/neural-networks-parameters-hyperparameters-and-optimization-strategies-3f0842fac0a5.

Yulia Gavrilova (2020): A Guide to Deep Learning and Neural Networks. Hg. v. Serokell. Online verfügbar unter https://serokell.io/blog/deep-learning-and-neural-network-guide, zuletzt aktualisiert am 2020.

8.2 Bilderverzeichnis

9 Anhang

9.1 Projektplan



9.2 Lokalisierung der Fussgänger Inseln des Datensets