HAW HAMBURG

MASTERTHESIS
Thomas Kanne-Schludde

Rekonstruktion der Kameraposition aus Fotos bekannter Objekte mit Deep Learning-Verfahren

FAKULTÄT TECHNIK UND INFORMATIK
Department Informatik

Faculty of Computer Science and Engineering
Department Computer Science

HOCHSCHULE FÜR ANGEWANDTE
WISSENSCHAFTEN HAMBURG
Hamburg University of Applied Sciences

Thomas Kanne-Schludde

Rekonstruktion der Kameraposition aus Fotos bekannter Objekte mit Deep Learning-Verfahren

Masterarbeit eingereicht im Rahmen der Masterprüfung
im Studiengang *Master of Science Informatik*
am Department Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer: Prof. Dr. Kai von Luck
Zweitgutachter: Prof. Dr. Andreas Meisel

Eingereicht am: 16. April 2021

Thomas Kanne-Schludde

Thema der Arbeit

Rekonstruktion der Kameraposition aus Fotos bekannter Objekte mit Deep Learning-Verfahren

Stichworte

Computer Vision, Machine Learning, Deep Learning, neuronale Netze, CNN, Bilderkennung, inhaltsbasierte Ortserkennung, Geoposition, Kameraposition, GPS-Bestimmung, Klassifikation

Kurzzusammenfassung

In dieser Arbeit wird eine „state-of-the-art“ Deep Learning Architektur trainiert, um die Kameraposition aus Fotos bekannter Objekte anhand der Pixel zu rekonstruieren. Der Ansatz wird als Klassifikationsproblem auf Basis synthetisch hergestellter Trainingsdaten definiert und anhand exemplarischer realer Testdaten die Machbarkeit evaluiert. Zudem erfolgt ein Vergleich mit der Verwendung realer Fotos als Trainingsgrundlage und weiterer Ansätze zur Bestimmung von Geopositionen.

Thomas Kanne-Schludde

Title of Thesis

Reconstructing the camera position from photos of known objects with deep learning methods

Keywords

Computer Vision, Machine Learning, Deep Learning, neural networks, CNN, image recognition, content-based image retrieval, visual place recognition, geoposition, camera position, GPS localisation, classification

Abstract

In this work, a state-of-the-art Deep Learning architecture is trained to reconstruct the camera position from photographs of known objects based on the pixels. The approach

is defined as a classification problem based on synthetically produced training data, and feasibility is evaluated using exemplary real test data. In addition, a comparison is made with the use of real photos as a training basis and other approaches for the determination of geo-positions.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einleitung

Durch die fortschreitende technische Entwicklung werden mobile Endgeräte immer leistungsfähiger und präsenter. Allein in Deutschland besitzen rund 86 % der Bevölkerung ein Smartphone (Stand 2020) [77]. Diese werden zudem vielfach bereits als Ersatz für professionelle Kameras genutzt, was in Kombination mit der zunehmenden Digitalisierung eine enorme Verfügbarkeit von Fotos im Internet zur Folge hat.

Fotos können technisch mit Metadaten wie Informationen über die verwendete Kamera oder Zeit und Ort der Aufnahme versehen werden. Beim „Geotagging“ wird bspw. die Geoposition der Kamera zum Zeitpunkt der Aufnahme als GPS-Koordinaten in Form von Längen- und Breitengrad in sogenannten EXIF[1]-Daten gespeichert [68]. Viele Online-Plattformen und Ergebnisse der Bildersuchen aus Suchmaschinen geben allerdings zumeist gar keine Geo-Informationen an und auch beim Versenden von Fotos über bekannte Messenger-Dienste werden diese Informationen in der Regel nicht mitgesendet.

Im Rahmen dieser Arbeit wird ein Mechanismus zur Korrektur und Ergänzung von EXIF-Informationen, im Speziellen der GPS-Informationen, gesucht. Exemplarische Untersuchungen [30] mit Diensten für Endkunden wie Googles Schnittstelle für maschinelles Sehen *Vision AI*[2] haben ergeben, dass bei einer Rückwärtssuche von Fotos[3] bekannter Sehenswürdigkeiten zwar häufig auch eine Geoposition bestimmt wird, diese in den meisten Fällen allerdings der Position des erkannten Objektes und nicht der Kameraposition entspricht. Darin begründet soll erörtert werden, ob sich die Geoposition der Kamera auch anhand der Pixel eines Bildes rekonstruieren lässt.

In der Forschung haben sich unterschiedliche Methoden zur Erkennung von Objekten in Bildern etabliert. Zum einen gibt es einige analytische Algorithmen aus dem Bereich der Computer Vision, die allerdings anfällig gegenüber Verdeckung der zu erkennenden

[1] Exchangeable Image File Format: http://exif.org (Letzter Zugriff: 25.03.2021)
[2] Google Vision AI: https://cloud.google.com/vision (Letzter Zugriff: 24.03.2021)
[3] Als Rückwärtssuche ist hier die Suche nach Informationen anhand eines Bildes als Sucheingabe gemeint.

Objekte sind und je nach Anwendungsfall schnell mit einem hohen Implementierungsaufwand verbunden sind. Mit der technischen Entwicklung und Verfügbarkeit leistungsfähiger Hardware hat sich zum anderen der Ansatz des maschinellen Lernens besonders mit *Deep Learning* basierten Methoden zur Bilderkennung aus dem Bereich der *Künstlichen Intelligenz (KI)* als vielversprechendes Mittel etabliert, Objekteigenschaften bei gegebener Robustheit gegenüber Verdeckungen zu erlernen. Die Rekonstruktion von Kamerapositionen wird folglich als Klassifikationsaufgabe mit Deep Learning Methoden definiert, woraus sich grundlegende Fragen, insbesondere in Hinblick auf die Verfügbarkeit und Qualität von Trainingsdaten, mit denen die KI trainiert wird, ergeben (denn eine KI ist nur so gut wie ihre Daten) [22]. Crowd-basierte Fotoplattformen bieten sich hier als Quelle für Trainingsdaten an, wobei Anzahl und Qualität der Daten stark variieren können. Das zu trainierende Modell benötigt Trainingsdaten mit korrekt annotierten GPS-Positionen, um daraus lernen zu können. Ist dies nicht gegeben (bspw. aufgrund einer unzureichenden Trainingsdatenmenge), müssten die Daten manuell nachgetragen werden, was einen großen zeitlichen Aufwand zur Folge hätte.

Ein weiterer Ansatz ist, die Fotos samt Metainformationen synthetisch zu generieren, womit der manuelle Aufwand der Annotation der Trainingsdaten entfällt. Ein von Tremblay et al. [69] entwickeltes Plugin für die Unreal Engine[4] bietet hier die Möglichkeit, 3D-Modelle aus verschiedenen Kameraperspektiven bei stark variierenden Bedingungen wie Lichtverhältnisse, Verdeckungen und Umgebungen zu simulieren und exportieren. Parallel werden Lage und Orientierung der aufgezeichneten Objekte im Raum und die Position der Kamera als Metadaten aufgezeichnet. Die Kamerapositionen werden in dieser Arbeit in GPS-Koordinaten umgerechnet, anschließend mit den exportierten Fotos trainiert und auf reale Testdaten angewendet.

1.1 Ziel der Arbeit

Mit dieser Arbeit soll die Verwendung von Deep Learning-Verfahren zur Rekonstruktion der Geoposition der Kamera anhand der Pixel eines einzigen Bildes erprobt werden. Die exemplarische geografische Domäne, innerhalb derer die Erkennung stattfinden soll, ist der Umkreis um eine bekannte Sehenswürdigkeit. Dabei soll herausgearbeitet werden, in welchem Maße die Verwendung synthetischer Trainingsdaten für die Erreichung des Ziels möglich ist, welche Komplikationen dies mit sich bringt und ob sie somit der Verwendung

[4] Unreal Engine: https://www.unrealengine.com (Letzter Zugriff: 19.03.2021)

realer frei zugänglicher Fotos aus dem Internet vorzuziehen ist. Zudem soll abschließend eine Einschätzung abgegeben werden, inwieweit sich dieses System zur Bereinigung von EXIF-Daten eignet und welche alternativen Verfahren möglicherweise besser geeignet wären.

1.2 Aufbau

Da die Experimente in dieser Arbeit auf Bilderkennung im Zusammenhang mit Deep Learning Methoden beruhen, beginnt sie mit einem Analyseteil, in dem Grundlagen aus dem Umfeld der Computer Vision über Machine Learning bis hin zu Deep Learning Architekturen aus dem Bereich *Convolutional Neural Networks (CNN)* behandelt werden. Dazu werden vergleichbare Arbeiten analysiert und die theoretische Basis für das weitere Vorgehen in den folgenden Kapiteln geschaffen. Da sich diese Arbeit maßgeblich mit der Beschaffung und Erstellung von Datensätzen beschäftigt, werden in Kapitel 3 verschiedene Datenquellen analysiert und Methoden, diese für das Training von CNN-Klassifikatoren aufzubereiten, vorgestellt. Darauf folgt mit Kapitel 4 der praktische Teil, in dem Datensätze auf Basis von Kapitel 3 erstellt, trainiert und analysiert werden. Die Evaluation (Kapitel 5) nimmt anschließend eine Bewertung der Experimente des praktischen Teils vor und gibt Aufschluss über mögliche Verbesserungen. Den Abschluss bilden schließlich eine gesamtheitliche Zusammenfassung der Arbeit und ein Ausblick auf mögliche nachfolgende Arbeiten.

2 Analyse

Wie kann man den Entstehungsort eines Fotos bestimmen, wenn dessen Metadaten darüber keinen Aufschluss liefern? Zur Beantwortung dieser Frage ist es naheliegend, dass der Inhalt des Bildes anhand dessen Pixel analysiert werden muss. Im Englischen spricht man bei einer inhaltsbasierten Suche nach Bildern vom *Content Based Image Retrieval (CBIR)* [84]. Damit ist gemeint, dass sich anstelle der Metadaten auf Farb- und Helligkeitswerte der Bildpunkte und markante Muster innerhalb eines Bildes gestützt wird.

Für den speziellen Fall der Bestimmung der Geopositionen eignen sich als Anhaltspunkte besonders Fotos, auf denen Sehenswürdigkeiten abgebildet sind, da deren Geopositionen in der Regel bekannt sind. Arbeiten, die sich mit diesem Ansatz beschäftigen, lassen sich häufig in die Kategorien „visuelle Ortserkennung“ [39] und „visuell-basierte Lokalisierung“ [51] einordnen, wobei erstere in der Regel darauf abzielen, die Umgebung zu erkennen und zweitere die exakte Pose der Kamera (Position und Orientierung) zu berechnen.

In diesem Abschnitt werden Beispiele beider Kategorien vorgestellt und zu Beginn eine Einführung in die Grundlagen der inhaltsbasierten Bildanalyse, auf die letztendlich alle Arbeiten aufbauen, gegeben. Diese beginnt mit analytischen Methoden aus dem Feld der Computer Vision und fokussiert sich ferner mit dem maschinellen Lernen auf lernbasierte Methoden insbesondere der Verwendung von tiefen neuronalen Netzen (Deep Learning). Es folgt eine Auseinandersetzung mit den Anforderungen an einen Datensatz für das Trainieren von Faltungsnetzen und in dem Kontext ein Vergleich zweier unterschiedlicher Herangehensweisen mit der Nutzung realer und synthetisch erzeugter Trainingsdaten.

2.1 Analytische Ansätze

Seit Jahrzehnten beschäftigt man sich schon mit der Entwicklung mathematischer Techniken für die Erkennung von Merkmalen aus den Texturen von Bildern. Die ersten Algo-

rithmen haben sich im Feld der Computer Vision entwickelt, denn bereits 1973 arbeiteten z. B. Fischler et al. [16] an Methoden, die Beschreibung eines visuellen Objektes innerhalb einer Fotografie wiederzufinden und damit die Gesichtserkennung voranzutreiben. Bei einer typischen inhaltsbasierten Bildsuche werden zu einem Anfragebild ähnliche Bilder im Datenbestand gesucht und das Bild, das nach einer Distanzberechnung der Eingabe am nächsten kommt, mit den angefragten Informationen ausgegeben. Dabei werden zuerst lokale Bildmerkmale (englisch: *Keypoint Features* oder *Points of Interest*) mit sogenannten *Feature Detector*-Algorithmen gesucht [84].

2.1.1 Feature-Detektoren und -Deskriptoren

Die Grundlage der Feature-Detektoren ist das Finden von Mustern und Merkmalen auf Basis berechneter Farbgradienten, die pixelweise anhand der umgebenen Pixel z. B. anhand von Farbintensitäten im Histogramm eines Bildes berechnet werden. Zwei wichtige Operationen sind die Erkennung von Kanten und Konturen, die mit Faltungsmatritzen berechnet werden (dazu später mehr in Abschnitt 2.2.2). Im Laufe der Zeit wurde aber eine Vielzahl unterschiedlicher Feature-Detektoren entwickelt, die in einem ausführlichen Vergleich von Funktionsweise und Leistung in der Arbeit von Mikolajczyk und Schmid [42] verglichen werden. Zum Finden und Vergleichen gleicher oder ähnlicher Merkmale in einer Menge von Bildern werden *Feature Descriptor*-Algorithmen eingesetzt, die jeweils die Region um die gefundenen Bildmerkmale herum beschreiben und deshalb bis zu einem gewissen Grad invariant gegenüber Skalierung, Orientierung und teilweise auch leichten Deformierungen der Merkmale, Bildrauschen und Belichtung sind. Mit Hilfe der *Hough Transformation* [75], einem bereits 1962 entwickelten Kreis- und Geradendetektor für Schwarz-Weiß-Bilder, können Kanten und Geraden auch dann erkannt werden, wenn sie unterbrochen sind oder durch andere Objekte verdeckt werden. Dadurch können die Fluchtpunkte von Objekten berechnet werden, was bei der Kalibrierung von Kameras und der Posenerkennung von Objekten hilfreich ist. Bekannte Vertreter dieser in einem Bild lokal operierenden Deskriptoren sind z. B. der von David G. Lowe vorgestellte *SIFT (Scale invariant feature transform)* [38] Algorithmus sowie dessen Weiterentwicklung *SURF (Speeded Up Robust Features)* [5] und die performante Alternative *ORB (Oriented FAST and Rotated BRIEF)* [54], die Informationen über die Orientierung der Objekte hinzufügt. Für den Vergleich einiger lokaler Deskriptoren kann hier analog zu [42] auf eine weitere Arbeit von Mikolajczyk und Schmid [41] verwiesen werden, in der auch ein experimenteller Performance-Vergleich stattfindet sowie auf eine aktuelle Analyse von Tareen

et al. [66]. Zusätzlich zu den lokalen wurden ebenfalls globale Deskriptoren wie Gist [46] entwickelt, die zwar performanter arbeiten können, da sie das Bild als gesamte Einheit analysieren, dafür allerdings anfälliger gegenüber Perspektivwechsel, Rauschen in den Daten und Verdeckung sind.

2.1.2 Anwendungsfälle

Die soeben beschriebenen Algorithmen sind bspw. Grundlage für das Erstellen von Panoramafotos auf Basis mehrerer einzelner Fotos [6] und werden für die Objektverfolgung in Videos [17] genutzt. Ein aktiv erforschtes Feld ist ferner die Rekonstruktion von 3D-Modellen anhand hunderter unstrukturierter Fotos aus dem Internet [21]. In der Arbeit „Photo Tourism“ von Snavely et al. [61] kommen Stereo-Matching-Algorithmen [58, 19] zum Einsatz, mit denen sich bspw. detaillierte 3D-Modelle ganzer Gebäudefassaden aus hunderten verschiedenartiger Fotos erzeugen lassen. Im Speziellen wird mit sogenannten *Structure-from-Motion*-Algorithmen gearbeitet, mit denen 2D-Merkmalspunkte iterativ aus Foto-Paaren verschiedener Perspektiven eines aufgenommenen Objektes extrahiert und daraus Tiefenkarten mit 3D-Koordinaten rekonstruiert werden. In Kombination mit Computer Vision Werkzeugen (Nutzung effizienter Feature Deskriptoren (SIFT), Indizierungs- und Vergleichsmethoden auf Basis von Entscheidungsbäumen etc.) können relative Kameraparameter wie Brennweite, Orientierung und Geoposition berechnet werden, womit diese Arbeit ein Vertreter der direkten visuellen Lokalisierung nach [51] ist. Die initiale Brennweite wurde aus EXIF-Informationen entnommen und absolute GPS-Positionen durch Abgleich mit Satellitenfotos berechnet. Diese Arbeit ist die erste dieser Art, in der unstrukturierte Fotos aus diversen Internetquellen genutzt wurden (Bildersuche in Suchmaschinen oder Crowd-basierte Fotoplattformen wie Flickr[5] etc.). Unter anderem wurden 2.635 Fotos der Notre Dame Kathedrale in Paris prozessiert und davon 597 anhand der Kameraparameter innerhalb der Szene registriert [61].

2.2 Maschinelles Lernen

Neben den analytischen Methoden hat sich mit dem maschinellen Lernen (engl. *Machine Learning*) eine Methode durchgesetzt, die durch lernende Algorithmen menschliche Intelligenz nachzuahmen versucht. Deshalb wird Machine Learning in vielen Publikationen

[5] https://www.flickr.com (Letzter Zugriff: 10.01.2021)

auch als Teilgebiet der künstlichen Intelligenz angesehen [9] und wird namentlich bereits 1958 in einer von Arthur Samuel bei IBM veröffentlichten Arbeit verwendet, in der gezeigt wird, dass Computer programmiert werden können, um Schach zu spielen [56]. Das Prinzip baut darauf auf, bestimmte Aufgaben anhand von großen Datenmengen durch Wiederholungen zu lernen. Der Algorithmus muss deshalb nicht explizit für die entsprechende Aufgabe programmiert werden, sondern kann seine Architektur kontinuierlich durch Erfahrungswerte anpassen und ändern. Diese Anpassungen finden durch mathematische Operationen auf Eingangsdaten statt, die dem Algorithmus zusammen mit erwarteten Ausgangsdaten übergeben werden. Prinzipiell wird hier versucht, eine möglichst genaue mathematische Funktion zu finden, die die Eingangsdaten auf die Ausgangsdaten abbildet. Dieser iterative Vorgang wird als *Training* bezeichnet und ermöglicht dem Algorithmus, mithilfe einer Fehlerfunktion (engl. *Loss*) zur Bewertung der Ergebnisse auch aus seinen Fehlern zu lernen. [13, 18]

Entscheidend für den Erfolg eines Trainings ist die Auswahl geeigneter Trainingsdaten und deren Eigenschaften, die häufig als *Feature Selection* [45, 7] bezeichnet wird (zur Erstellung von Trainingsdaten folgt später mehr in Kapitel 3). Ein typischer Anwendungsfall für die Verwendung maschinellen Lernens ist die Klassifikation von Objekten. Am Beispiel der Bildverarbeitung betrachtet, soll der Algorithmus bspw. aus einer Menge von Fotos Hunde und Katzen erkennen, unterscheiden und kategorisieren können, auch wenn es sich um individuelle Fotos, die unter völlig unterschiedlichen Bedingungen erstellt wurden, handelt. Der Algorithmus muss also generalisieren können, wofür sich wiederholende Muster in den Trainingsdaten gefunden werden, anhand derer dann eine Bestimmung erfolgt. Der Trainingseffekt setzt dabei durch die Verknüpfung der entsprechenden Trainingsbilder mit den Ausgabewerten, die hier die Klasse (Katze, Hund) beschreiben, ein. [18]

Die Bestimmung von kontinuierlichen Zielwerten bezeichnet man als Regression, während im Falle der Klassifikation eine diskrete Zielmenge definiert wird. Ziel ist, die Klassen für Eingabefotos bestimmen zu können, die der Algorithmus beim Training noch nicht gesehen hat. Für das Training muss deshalb eine hinreichend große Menge von Ein- und Ausgabedaten mit korrektem Funktionswert zur Verfügung stehen. Um die Güte eines trainierten Modells anschließend durch Tests beurteilen zu können, stehen unterschiedliche Metriken zur Verfügung. Im Falle einer Klassifikation ist die einfachste Angabe die *Genauigkeit* (engl. *Accuracy*). Diese gibt den Anteil der richtigen Vorhersagen aus allen getroffenen Vorhersagen in Prozent an. Mit der *Precision* wird der Anteil der richtigen

positiven Vorhersagen aus allen positiven Vorhersagen bestimmt, während der *Recall* diese aus der Gesamtheit der tatsächlich positiven Ergebnisse berechnet [62, 22].

Dieses Beispiel fällt damit in die Kategorie *überwachtes Lernen* (engl. *Supervised Learning*). Beim *unüberwachten Lernen* (engl. *Unsupervised Learning*) hingegen werden die Eingangsdaten nicht markiert, sodass der Algorithmus lediglich darin verborgene Strukturen zu erkennen versucht. Eine Bewertung mithilfe einer Fehlerfunktion wie beim überwachten Lernen ist hier nicht möglich. Es wird eher dazu verwendet, um große Datenmengen zu gruppieren und kommt z. B. häufig in Onlineshop-Systemen zur Analyse und Steuerung des Kaufverhaltens der Kunden zum Einsatz.

In dieser Arbeit wird das Supervised Learning als Trainingsgrundlage verwendet. Für einen detaillierten Vergleich unterschiedlicher Trainingsmethoden, u. a. die Kombination aus beiden genannten oder dem *Reinforcement Learning*, bei dem das trainierende Modell durch Belohnung oder Bestrafung von außen lernt, bietet die Arbeit von Sathya et al. [57] einen guten Ausgangspunkt.

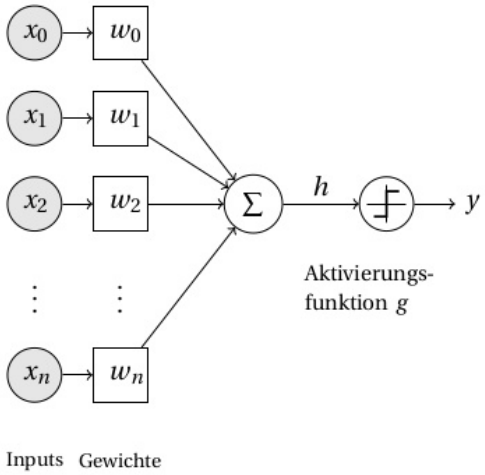
2.2.1 Neuronale Netze

Künstliche neuronale Netze haben sich für besonders komplexe Anforderungen als Teilgebiet der künstlichen Intelligenz innerhalb des maschinellen Lernens durchgesetzt. Die Entwicklung geht auf die Erfindung des Perzeptron 1958 (ein Neuron mit Eingang, Gewichtung und Ausgang) [53] zurück und basiert in Teilen auf neurobiologische Hypothesen zur Informationsverarbeitung in Nervenzellen von Säugetieren, die Eingangssignale nach Erreichen eines kritischen Schwellenwertes weitergeben. Mit der Verkettung mehrerer künstlicher Neuronen zu einem mehrschichtigen Perzeptron (engl. *Multilayer Perceptron (MLP))* lassen sich äußerst komplexe Kombinationen parallel verteilter nichtlinearer Berechnungsmodelle erstellen, mit denen im Prinzip jede stetige mathematische Funktion angenähert werden kann und sich somit diverse Datenmengen beschreiben lassen. Zwischen einer Eingangsschicht und einer Ausgangsschicht befinden sich sogenannte verdeckte Schichten (engl. *Hidden Layer*), deren Bezeichnung daher rührt, dass ihre Funktionsweisen nicht dargestellt werden und sie keine realen Parameter besitzen. Bei der Verwendung mehrerer verdeckter Schichten spricht man von tiefen neuronalen Netzen bzw. *Deep Learning*, die zunehmend abstraktere Merkmale aus den Eingabedaten extrahieren.

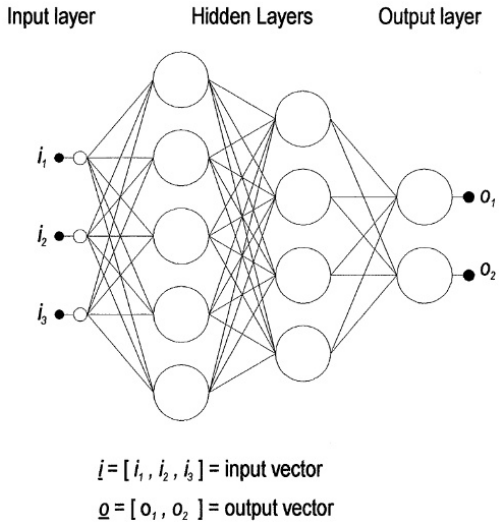
Auch wenn dieses Teilgebiet des maschinellen Lernens bereits sehr alt ist, kann das Potenzial dieser Architekturen erst seit einigen Jahren durch die technische Weiterentwicklung von Speicherressourcen und Rechenleistung effektiv genutzt werden. [22]

Abbildung 2.1a zeigt ein einzelnes Neuron, an dem die Eingangssignale x_0 bis x_n anliegen. In der mathematischen Funktion werden sie als Vektor repräsentiert und dann mit w_0 bis w_n gewichtet. Die Summe der gewichteten Eingänge wird dann der Schwellenwertfunktion g, die im Kontext neuronaler Netze *Aktivierungsfunktion* genannt wird, übergeben. Wenn diese über dem Schwellenpotential liegt, wird das Neuron „aktiv“, sodass die Information an die nachfolgende Schicht weitergegeben wird. Bekannte Aktivierungsfunktionen sind z. B. die logistische Funktion (engl. *Sigmoid*) oder *Softmax*-Funktion.

Abbildung 2.1b zeigt ein künstliches neuronales Netz mit vier Schichten (Eingangsschicht mit den Eingangssignalen i_1 bis i_3, zwei verdeckte Schichten und eine Ausgangsschicht mit den Ausgangssignalen o_1 und o_2). Wenn jedes Neuron einer Schicht mit den Neuronen der folgenden Schichten verbunden ist, nennt man dies eine *Fully Connected* Schicht. Dies ist mindestens in der Ausgabeschicht der Fall, die die berechneten Zielwerte ausgibt und im Falle eines Klassifikators einen Ausgang pro zu bestimmender Klasse enthält (in diesem Beispiel die Ausgänge o_1 und o_2).



(a) Ein künstliches Neuron mit Eingängen, Gewichten, Aktivierungsfunktion und einem Ausgang (aus [18])



(b) Multi-Layer-Perceptron (aus [20])

Abbildung 2.1: Einzelnes Neuron und Verkettung von Neuronen in einem Multi-Layer-Perceptron

Das Ziel des Trainings ist es, die Ausgabewerte möglichst akkurat zu bestimmen. Dafür werden sie mit den ursprünglichen Markierungen der Daten verglichen und daraus der Fehler (engl. *Loss*) berechnet. Dieser hängt nur von den Gewichten ab, die es somit zu optimieren gilt. Dafür werden Optimierungsalgorithmen verwendet, mit denen das Minimum der Fehlerfunktion berechnet wird. Dabei kommt häufig das Gradientenabstiegverfahren zum Einsatz, mit dem versucht wird, die Minima einer Funktion zu ermitteln. Der Vorgang des Anpassens der Gewichte mithilfe von Optimierungsalgorithmen wird als *Backpropagation* bezeichnet. Eine detaillierte Beschreibung zur Funktionsweise verschiedener Optimierungsverfahren lässt sich u. a. in Goodfellow et al. [22] finden. [22, 18]

Da es in dieser Arbeit um die Bilderkennung mit neuronalen Netzen geht, wird im nächsten Abschnitt näher auf die speziellen Faltungsnetze eingegangen.

2.2.2 Convolutional Neural Networks

Allgemeiner Aufbau

Faltungsnetze (Convolutional Neural Networks) haben ihren Ursprung bereits 1989 in LeCun [35] und gehören zu den besten Lernalgorithmen für das Verstehen von Bildinhalten [37]. Sie sind eine spezielle Form der künstlichen neuronalen Netze, die ein angepasstes Verhalten für die Verarbeitung von Eingabedaten haben, die sich als Gitterstruktur darstellen lassen. Sie lassen sich dadurch optimal für die Bilderkennung nutzen, da ein Bild als zweidimensionale $n * n$-Matrix dargestellt wird, dessen Zahlenwerte in der Regel im Bereich von 0 bis 255 für die Intensität des Farbwertes eines Pixels angegeben werden. Handelt es sich um ein farbiges Bild, wird pro Farbkanal eine Matrix aufgespannt. Mit klassischen MLPs könnten auch Bilderkennungsaufgaben durchgeführt werden, was häufig anhand von Trainings mit dem MNIST-Datensatz[6] gezeigt wird. Tests eines klassischen MLP mit eigenen nicht ganz sauber handgeschriebenen Zahlen können allerdings schnell zu deutlich geringeren Ergebnissen führen, da diese eine hohe Empfindlichkeit gegenüber Translation, Rotation und Skalierungen von Bildern haben [18].

CNNs unterscheiden sich zu herkömmlichen MLPs dadurch, dass sie mindestens eine Schicht mit Faltungsoperationen – den *Feature Maps* - besitzen (siehe Abbildung 2.2). Faltungsoperationen wurden bereits in den in Abschnitt 2.1 vorgestellten analytischen

[6] Datensatz mit handgeschriebenen Zahlen mit ca. 60.000 Trainingsbeispielen: http://yann.lecun.com/exdb/mnist (Letzter Aufruf: 23.03.2021)

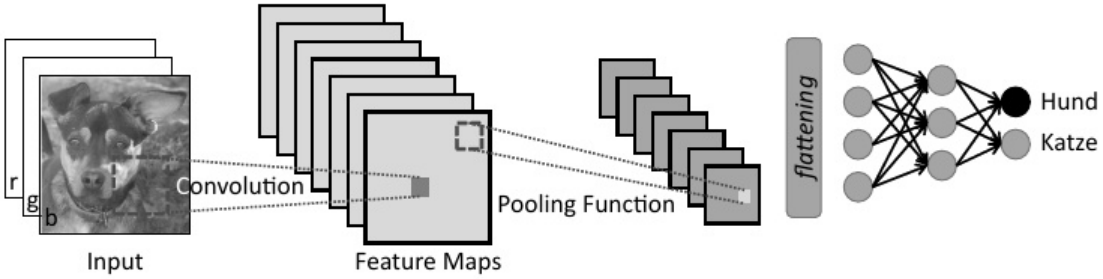


Abbildung 2.2: Modell eines Convolutional Neural Network (aus [18])

Algorithmen verwendet, sodass CNNs prinzipiell als Nachfolger dieser Methoden für komplexe Fragestellungen im Machine Learning Bereich angesehen werden können. Sie haben die Eigenschaft, dass damit lokale Merkmale der Bilder extrahiert werden können und können somit z. B. als Kantendetektor eingesetzt werden. Die Realisierung erfolgt über quadratische *Faltungsmatritzen* (auch als *Kernel* bezeichnet) meist ungerader Abmessung, die iterativ auf alle Bildbereiche angewandt werden, indem sie die Werte des Bildes mit denen des Kernels multiplizieren und aufsummieren. Das Ergebnis wird anschließend in eine Feature Map eingetragen (siehe Abbildung 2.3). Die Werte eines Kernels haben somit die Funktion von Gewichten, wie sie bei den Neuronen verwendet werden und werden dementsprechend trainiert. Dadurch müssen deutlich weniger Freiheitsgrade trainiert werden, als dies bei klassischen MLPs der Fall wäre, wo jeder Eingang (Pixel) mit jedem Neuron verbunden ist. Durch die Faltung können lokale Zusammenhänge der Merkmale in den Bildern erkannt werden. Zudem teilen sich alle Punkte der Feature Maps die gleichen Gewichte, sodass bestimmte Merkmale unabhängig von der Position innerhalb des Bildes erkannt werden können (Robustheit gegenüber Translation) und gleichzeitig Speicherplatz gespart wird. Mit zusätzlichen *Pooling Layers* (vgl. Abbildung 2.2) werden nur die relevantesten Signale an die nächsten Schichten weitergegeben. Diese Informationsverdichtung erhöht die Abstraktion der Daten und sorgt für bessere Generalisierung der Vorhersagen. Mit dem *Maxout Pooling* lässt sich z. B. eine höhere Invarianz gegenüber sich ändernder Lichtverhältnisse erreicht werden [22]. Wie bei den klassischen MLPs können auch bei den CNNs mehrere Schichten hintereinander gehängt werden. Je tiefer das Netz wird, desto globaler werden die ermittelbaren Zusammenhänge. Mit der *Flattening*-Schicht wird schließlich wieder eine voll vernetzte Schicht erzeugt, die dann die Eingabewerte für ein klassisches MLP bilden, das dann die Vorhersagen trifft.

Die Faltungsschicht erzeugt damit prinzipiell die Features für das MLP. Im Gegensatz zu den analytischen Ansätzen sind durch CNNs damit keine separaten Algorithmen für

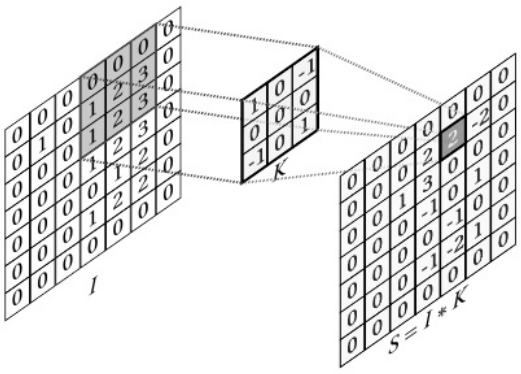


Abbildung 2.3: Faltungsoperation mit $3 * 3$-Kernel (aus [18])

die Extraktion von Merkmalen mehr nötig, was ein dynamisches Lernen von Rohdaten ermöglicht [32, 1, 83]. Sie dienen im Bereich der inhaltsbasierten Bildsuche deshalb häufig auch als Werkzeug zur Erzeugung der Repräsentationen der Bilder für den CBIR-Prozess, die aus den Feature Maps der Zwischenschichten entnommen werden [40]. Dabei zeigt sich bei der Verwendung ein bedeutender Performance-Vorteil gegenüber klassischen analytischen Methoden [32, 34]. Die Vorhersagen werden in weiteren Arbeiten [80, 11] je nach Anwendungsfall als mit menschlicher Performance vergleichbar und teilweise besser eingestuft, unter der Voraussetzung, dass die Repräsentationen bei ausreichender Menge an Daten gelernt wurden. Für die Bestimmung einzelner Bilder wird bei der Verwendung neuronaler Netze zudem keine Datenbank benötigt, um das Anfragebild abgleichen zu können. Stattdessen werden die Vorhersagen durch die trainierte Zielwertfunktion des Modells direkt getroffen und ist in der Regel sehr speicherschonend [59].

Die manuelle Arbeit der Merkmal-Extraktion bei den analytischen Methoden wird durch CNNs folglich durch die automatische Extraktion ersetzt und beläuft sich stattdessen eher auf das Design der CNN Architektur selbst.

Architekturen

Generell gibt es eine Vielzahl anpassbarer Variablen, Parameter und sogenannter Hyperparameter zum Optimieren des Lernverhaltens von neuronalen Netzen. Speziell im Fall der CNNs kommen einige hinzu, wie die

- Anzahl der Faltungsschichten,

- Anzahl, Größe und Faltungsoptionen der Kernel und
- Anzahl und Art des Pooling.

Aus diesem Grund ist eine Vielzahl neuer Optimierungsstrategien, Design-Muster etc. entstanden, die z. B. in Machine Learning Wettbewerben auf ihre Anwendbarkeit geprüft werden. Speziell im Bereich der Bilderkennung gibt es die *ImageNet Large Scale Visual Recognition Challenge (ILSVRC)*[7] oder Wettbewerbe auf der Machine Learning Plattform *Kaggle*[8]. In diesen Wettbewerben überwiegen zumeist Deep Learning Ansätze, die größenteils mit unstrukturierten Daten arbeiten, wozu auch die Bilderkennung gehört. Khan et al. [32] analysieren in ihrem umfangreichen Survey-Paper viele der gängigen CNN-Architekturen, auf das für ein tiefergehendes Verständnis deshalb an dieser Stelle verwiesen sei.

Als besonders performant hat sich u. a. die *Inception Architecture* - der Gewinner des ILSVRC 2014 - erwiesen. Die erste Version des als Klassifikator eingesetzten Modells wurde 2014 von Szegedy et al. [64] bei Google veröffentlicht und seitdem laufend weiterentwickelt [65].

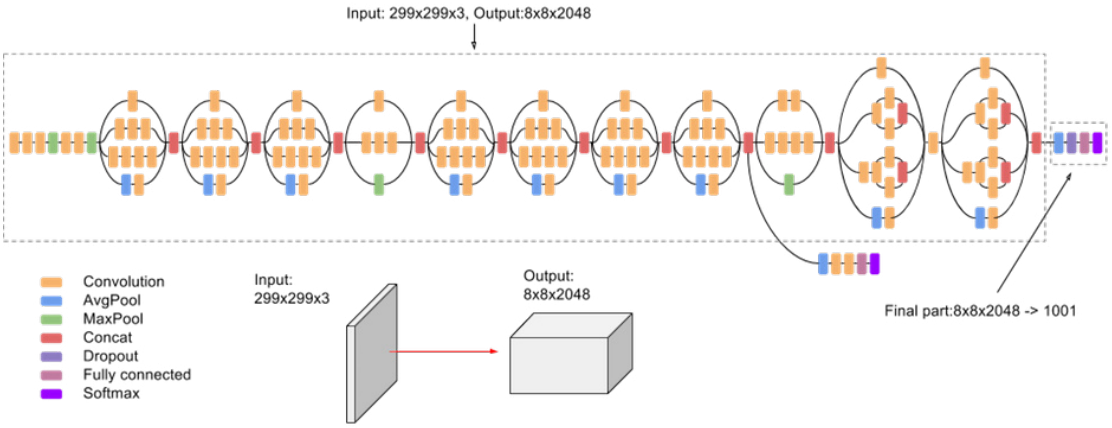


Abbildung 2.4: Schematische Darstellung der InceptionV3-Architektur (aus [12])

Die wesentliche Besonderheit ist das *Branching* innerhalb einer Schicht in sogenannte Inception-Blöcke (vgl. High-Level Schema der *InceptionV3*-Architektur in Abbildung 2.4). Diese enthalten unterschiedliche Convolutional-Layer nebeneinander und fassen deren Ausgaben mit den *Concat*-Bausteinen wieder zusammen. Laut der Autoren zeichnet sich

[7] http://www.image-net.org/challenges/LSVRC (Letzter Zugriff: 23.03.2021)
[8] https://www.kaggle.com (Letzter Zugriff: 23.03.2021)

dadurch die Architektur besonders durch das gute Lokalisieren und Erkennen von Objekten in Bildern aus. Es konnte vortrainiert auf dem ImageNet-Datensatz[7], der 1.281.167 Millionen Trainingsfotos mit 1.000 Klassen enthält, eine Genauigkeit von 78 % vorweisen. Weiter kommt es mit deutlich weniger trainierbaren Parametern aus als ähnliche Modelle wie z. B. VGG19 oder AlexNet [65]. Aufgrund der guten Eigenschaften und der Tatsache, dass die InceptionV3-Architektur in einem Paper von Weyand et al. [78] verwendet wurde, das dieser Arbeit als Inspiration diente, wurde es auch hier verwendet. Das Modell steht zudem innerhalb der Deep Learning API Keras zur Verfügung[9], die für die Implementierung verwendet wird (siehe Abschnitt 4.1.1) und kann ebenso wie viele weitere der gängigen Architekturen mit auf dem ImageNet-Datensatz vortrainierten Gewichten verwendet werden. Vortrainierte Modelle lassen sich häufig auch sinnvoll auf Anwendungsfälle anderer Domänen anwenden, wodurch Trainingsdauer eingespart und insgesamt bessere Trainingsergebnisse erreicht werden können. Dieses Vorgehen wird auch als *Transfer Learning* [47] bezeichnet.

2.3 Bestimmung von Geokoordinaten als Klassifikationsproblem

Im vorherigen Abschnitt wurde die Nutzung von CNNs zur Herstellung der Repräsentationen für die inhaltsbasierte Bildsuche angesprochen. Die ersten bedeutenden Erfolge in der Bilderkennung haben sie aber mit der Klassifikation erreicht [64]. Es gibt daher auch im Bereich der inhaltsbasierten Bildsuche Ansätze, die Repräsentationen mithilfe eines Klassifikators zu erzeugen und die zu trainierenden Bilder entsprechend „nur“ mit Klassen-Labels zu annotieren, was sich positiv auf die Skalierbarkeit bei großen Datensätzen auswirkt [8]. Es gibt allerdings auch Arbeiten, die die Aufgabe der visuellen Ortserkennnung (vgl. Einführung dieses Kapitels) insgesamt als Klassifikationsproblem definieren. Da in dieser Arbeit die Geoposition der Kamera eines Bildes bestimmt werden soll, wird unter dessen Berücksichtigung in diesem Abschnitt noch einmal der Bogen von analytischen Methoden zur Nutzung der Faltungsnetze durch entsprechende Beispiele gespannt.

Zur Erinnerung: In der im Abschnitt 2.1.2 vorgestellten Arbeit *Photo Tourism* [61] war die Bestimmung von GPS-Koordinaten ein Nebenprodukt der eigentlichen Zielsetzung

[9] Übersicht der mit der Keras API verwendbaren vortrainierten Modelle: https://keras.io/api/applications (Letzter Aufruf: 23.03.2021)

und wurde relativ zu bekannten oder bereits berechneten Daten approximiert. Der Fragestellung, wie man GPS-Koordinaten aus einem einzelnen Bild bestimmen kann, gehen Hays und Efros mit *IM2GPS* [23] allerdings explizit nach und versuchen mit Computer Vision Algorithmen die GPS-Positionen von Fotos auf einer weltweiten Skala mit über sechs Millionen weltweit verstreuten GPS-getaggten Fotos zu bestimmen. Die Entwicklung erfolgte allerdings auf Kosten einer sehr komplexen Pipeline mit diversen Filterkriterien auf den Trainingsdaten. Diese umfassten:

- eine Kombination von sechs verschiedenen globalen Feature-Deskriptoren (siehe [46]),
- die Berechnung von Farb- und Texturhistorgrammen, Clustering in Kategorien wie „Boden“, „Himmel“ etc.,
- Erzeugung von *Bag-of-visual-Words* [60] mit SIFT Features und anschließender
- Bestimmung der Geoposition durch nächste Nachbarn mit anschließendem Clustering und Training von Support Vector Machines (SVM) [24].

Eine Arbeit, die darauf aufbaut, aber statt analytischer Methoden CNNs verwendet, ist das von Weyand et al. bei Google entwickelte *PlaNet*-Projekt [78] und damit der erste Klassifikator auf einer globalen Skala. Die Klassen werden anhand eines Rasters, das die Erde in unterschiedlich große Zellen einteilt, gebildet. Die Größe der Zellen richtet sich dabei nach der Anzahl der in dieser Region vorhandenen Trainingsfotos. Bei Überschreiten einer definierten Anzahl von Fotos pro Zelle wird diese Zelle geviertelt (dazu später mehr in Abschnitt 3.2). Insgesamt wurden 91 Millionen Trainingsfotos mit GPS-Koordinaten aus allen denkbaren Internetquellen gewonnen und 2,5 Monate lang auf 200 CPU-Kernen trainiert. Das Trainings-Modell basiert auf der InceptionV3-Architektur mit einem abschließenden Softmax-Layer zur Klassifikation von 26.263 Zellen bzw. Klassen, für die bei der Bestimmung Vorhersage-Wahrscheinlichkeiten, also die Sicherheit in Prozent, mit der es sich bei der Schätzung um einen Treffer handelt, angegeben werden.

Durch die Unterteilung der Zellen wird eine Ausbalancierung der Trainingsdaten auf die Klassen angestrebt. In Bereichen mit schlechter Abdeckung der Daten sorgt dies aufgrund großer Klassen allerdings für eine Begrenzung der Genauigkeit der Klassifikation. Umgekehrt würde eine Erhöhung der Granularität an diesen Stellen kleinere Klassen mit weniger Trainingsdaten erzeugen und durch Erhöhung der Gesamtzahl der Klassen die zu trainierenden Parameter erhöhen. Dies wirkt sich negativ auf Skalierbarkeit und Generalisierbarkeit aus. Mit einer Erweiterung [59] versuchen die Autoren diesem durch eine

Kombination mehrerer Klassifikatoren zusätzlich erzeugter Überlappungen des Zellen-Netzes entgegenzuwirken. Letztendlich erreichen sie auf Straßen-Ebene im Umkreis von 1 km eine Genauigkeit von 16,5 %, im Umkreis von 100 km 42,6 % und sogar 81,9 % im 2.500 km Umkreis.

Mit einer Erweiterung des IM2GPS Projektes haben Vo et al. [76] weiterhin eine Kombination aus dem ursprünglichen Image Retrieval Verfahren und Deep Learning Methoden zum Vergleich mit dem PlaNet-Projekt angestrebt. Sie konnten hier zwar nicht die Ergebnisse des PlaNet-Projektes übertreffen, konnten aber Performance Vorteile des Image Retrieval Verfahrens gegenüber eines Klassifikators bei feiner Granularität beobachten. Beide Projekte bestimmen die Geopositionen allerdings erst ab einem Umkreis von einem Kilometer und wenden folglich keine Datensätze zum detaillierten Training eines einzigen Objektes an. Da dies jedoch Bestandteil dieser Arbeit sein soll, folgt deshalb im nächsten Abschnitt die Auseinandersetzung mit Datensatz-Anforderungen und Quellen.

2.4 Datensätze

Für die Experimente in dieser Arbeit werden eigene Datensätze zum Trainieren des Deep Learning Modells verwendet. Für die Erzeugung sind im Vorfeld einige Punkte zu beachten, die in diesem Abschnitt erläutert werden sollen.

2.4.1 Anforderungen an einen Datensatz

Generell gilt für das Training von Machine Learning Algorithmen, dass eine ausreichende Menge Beispieldaten vorhanden sein muss, um zufriedenstellende Ergebnisse zu erhalten. Wie viele Daten genau damit gemeint sind, kann nicht verallgemeinert werden, da dies stark vom Anwendungsfall, der verwendeten Architektur und der Qualität der Daten abhängt. Da in diesem Projekt ein Training mit einem Convolutional Neural Network und eigenen Datensätzen durchgeführt wird, soll hier kurz einführend erläutert werden, worauf dabei in diesem Kontext zu achten ist.

Aufbau der Trainingsdaten

Damit Bilder als Eingangsdaten für Machine Learning Algorithmen verwendet werden können, müssen sie in eine Gitterstruktur gebracht werden, in der jede Zelle die Inten-

sität des Farbwertes jedes Pixels von 0 bis 255 angibt. Diese Gitterstruktur liegt als zweidimensionaler Vektor vor, von denen im Falle eines RGB-Bildes mit drei Farbkanälen folglich drei pro Bild als Eingabe verwendet werden. Die CNN-Architekturen bieten eine Eingangsschicht, sodass Bilder nach einer Umwandlung mit gängigen Softwarebibliotheken wie OpenCV[10], scikit-learn[11] oder Keras direkt verwendet werden können. Zum Lernen von Informationen aus den Eingabebildern müssen diese jeweils entsprechend mit Labels annotiert werden. Dabei ist es wichtig, dass die Informationen sogenannte *Ground Truth* Daten sind, womit gemeint ist, dass diese definitiv korrekt sein müssen. Ansonsten würde der Algorithmus Vorhersagen auf Basis falscher Informationen treffen, womit die Fehlerquote wiederum vestärkt und somit das Ziel verfehlt würde. Bei der Annotation von Bildern kann zwischen einem und mehreren Objekten unterschieden werden. Sollen gleichzeitig mehrere Objekte innerhalb eines Bildes erkannt werden können, können Begrenzungsrahmen (engl. *Bounding Boxes*) um die Objekte gezeichnet werden, deren Koordinaten der jeweiligen Eckpunkte zusammen mit dem Namen der Objektklasse in einer separaten Datei gespeichert werden. Diese Eckpunkte werden schließlich für jedes Objekt zum Trainieren des Algorithmus verwendet. Alternativ kann auch das gesamte Bild mit einem Label wie z. B. „Hund“ oder „Katze“ versehen werden. Im Falle der Bestimmung von GPS-Koordinaten können beide Verfahren nützlich sein. Die im vorherigen Abschnitt vorgestellte Arbeit PlaNet [78] geht den Weg einer einmaligen Annotation des ganzen Bildes mit einer GPS-Koordinate. Im folgenden Abschnitt 2.5 wird schließlich die Bounding Box Variante anhand einer weiteren Arbeit vorgestellt. An dieser Stelle wäre somit wieder relevant, ob das zu trainierende Modell als Klassifikator oder Regressionsmodell umgesetzt werden soll (vgl. Kapitel 2). [18, 22]

Hardwareanforderungen

Das Training komplexer Netzarchitekturen besonders im Zusammenhang mit Bilddaten kostet viel Rechenleistung. Insbesondere wird GPU-Rechenleistung benötigt, da Grafikkarten für die Matrix-Vektor-Berechnungen ausgelegt sind. Ausschlaggebend ist neben der Anzahl der Trainingsbilder dessen Auflösung. Der MNIST-Datensatz besteht z. B. aus 28 x 28 Pixel kleinen Grauwert-Fotos und entspricht damit 784 Merkmalen pro Foto. Die Menge von 60.000 Trainingsfotos dieses Datensatzes ist in der Regel für jegliche aktuelle Privatrechner ohne Probleme zu trainieren. Nutzt man stattdessen 256 x 256

[10] OpenCV: https://opencv.org (Letzter Zugriff: 07.04.2021)
[11] scikit-learn: https://scikit-learn.org (Letzter Zugriff: 07.04.2021)

Pixel große RGB-Fotos, sind dies schon 196.608 Merkmale pro Foto. Die Wahl der genutzten Architektur sollte demnach auch aus Performance-Gesichtspunkten erfolgen. In Kapitel 4 wird die Auswirkung der Auflösungserhöhung getestet, für die Vorbereitungen eventueller Komplikationen mit weiteren Systemkomponenten (Größe des RAMs, Schreibgeschwindigkeit der Festplatten etc.) getroffen werden mussten.

Aufteilung in Trainings-, Validierungs- und Testdaten

Damit die gewählte Netzarchitektur und die genutzten Trainingsdaten während des Trainings auf ihre Funktionalität und Leistung hin geprüft werden können, teilt man den Datensatz in eine Trainings- und Validierungs-Submenge ein. Typischerweise erfolgt die Aufteilung in 80 % Trainingsmenge und 20 % Validierungsmenge. Da die Validierungsdaten nicht Bestandteil des Trainings sind, können sie nach jeder Epoche genutzt werden, um die Genauigkeit und den Fehler gegen den aktuellen Stand des trainierten Modells zu berechnen und mit Genauigkeit und Fehler der Trainingsdaten zu vergleichen. Die Validierungsergebnisse haben zwar keine Auswirkung auf die Anpassung der Gewichte während des Trainings, daran lässt sich aber das Verhalten des Netzes beobachten und ermöglicht gegebenenfalls ein rechtzeitiges Eingreifen in das Training. Wird nun ein allgemein hoher Trainings- und Validierungsfehler beobachtet, spricht man von einem *Underfitting*. Das bedeutet, dass das Netz die trainierten Daten nicht ausreichend lernt und deutet häufig auf eine zu geringe Komplexität zur Lösung der mathematischen Funktion hin. Divergieren beide Werte auseinander und entsteht zwischen ihnen eine große Differenz, handelt es sich um ein *Overfitting*, einer Überanpassung des Modells an die Trainingsdaten. Anders ausgedrückt bedeutet es, dass das Netz schlecht gegenüber neuen ungesehenen Daten generalisieren kann. In dem Fall sollte das Training gestoppt und je nach Ergebnis eine Anpassung der Architektur durch Verändern der Hyperparameter, Anwendung von Regularisierungsmethoden zur Verringerung der Netzkomplexität, Austausch der Netzarchitektur oder eine Überarbeitung der Trainingsdaten erfolgen. [18, 22] Für eine Überprüfung der tatsächlichen Praxistauglichkeit eines trainierten Modells können ferner Testdaten verwendet werden, die der Domäne entstammen, innerhalb derer das Modell verwendet werden soll. Diese werden im Anschluss des Trainings gegen das Modell geprüft und die Leistung an der Erkennungsrate gemessen. In dieser Arbeit folgt im Abschnitt 4.1.3 die Erstellung eines exemplarischen Testdatensatzes.

2.4.2 Datenquellen

Die Erfolgschancen beim Training eines neuronalen Netzes hängen maßgeblich von den Daten ab, mit denen es trainiert wurde. Die Beschaffung und Aufbereitung stellt für viele Fragestellungen eine große Herausforderung dar, die natürlich je nach Anwendungsanforderung sehr unterschiedlich ausfallen können. Da das Ziel dieser Arbeit die Erkennung von Geopositionen aus Bildern beinhaltet, folgt in diesem Abschnitt nun ein Überblick über mögliche Datenquellen für Fotos von Sehenswürdigkeiten mit angegebenen Geopositionen, die für das Training als Label verwendet werden sollen. Eine Möglichkeit wäre nun natürlich, eigene Fotos vor Ort manuell mit einer Kamera oder zusätzlich mit einer Drohne anzufertigen. Da dieses Vorgehen allerdings den Kosten-Nutzen-Rahmen sprengen und die Anwendbarkeit des Projektes auf ein einziges Objekt beschränken würde, soll im Folgenden ausschließlich auf im Internet bereits verfügbare Datenquellen zurückgegriffen werden.

Internetfotos

Als „Internetfotos“ werden hier Quellen für einzelne unsortierte Fotos zusammengefasst. Darunter fallen u. a. Ergebnisse der Bildersuchen aus gängigen Suchmaschinen wie z. B. Google oder Bing. Die Ergebnismengen enthalten allerdings extrem diverse Bildtypen und Motive. Diese lassen sich zudem nicht für die Erstellung eines gelabelten Trainingsdatensatzes nutzen, da den Ergebnissen keine GPS-Koordinaten mitgeliefert werden oder den entsprechenden Quellen nachgegangen werden muss, um dann eventuell Zugang zu Metadaten zu bekommen. Social Media Plattformen wie Facebook, Twitter, Instagram etc. beherbergen in den Posts der Nutzer:innen zwar eine Vielzahl von Fotos, die theoretisch verwendet werden könnten. Diese sind allerdings nur zum Teil öffentlich einsehbar (z. B. über die Suche nach Hashtags) und enthalten meistens keine Geotags. Zudem sind die Daten sehr unstrukturiert und können offiziell nicht über die entsprechenden APIs gefiltert und heruntergeladen werden.

Stattdessen bieten sich Crowd-basierte Plattformen wie Flickr oder 500px[12] an. Der Fokus liegt hier auf dem Bereitstellen von (semi-)professionellen Fotografien durch ihre Mitglieder:innen und bietet auch die Möglichkeit, intrinsische und extrinsische Kameraparameter eines Fotos anzugeben, wenn verfügbar und gewünscht. Flickr bietet nach eigenen Angaben eine Auswahl mehrerer Milliarden lizenzfreier Fotos. Eine Suche mit

[12] https://500px.com (Letzter Zugriff: 10.01.2021)

dem Begriff „Notre Dame“ führt bspw. zu 1,5 Millionen Treffern, eine Suche nach „Notre Dame Paris“ immerhin 500.000. Mit der Flickr API[13] lassen sich programmatisch über einige Filterkriterien große Datenmengen mit den entsprechend gewünschten Metadaten herunterladen. Dieser Umstand hatte Goesele et al. [21] bereits 2007 dazu motiviert, die Geometrie der Welt mit der größten verfügbaren Datensammlung zu rekonstruieren (siehe Abschnitt 2.1) und auch andere Arbeiten [61, 23, 78] nutzten Flickr als Primärquelle.

Allerdings entspricht die Nutzung dieser freien Daten verständlicherweise keinen Laborbedingungen: Die Fotos werden unter unzähligen unterschiedlichen Bedingungen in Bezug auf Bildqualität, Auflösung, Tageszeit, Wetterbedingungen, Lichtverhältnisse, Aufnahmeperspektive und -orientierung aufgenommen. Dazu enthalten viele Fotos Verdeckungen der aufgenommenen Motive durch Personen, Pflanzen, Autos und weiteren Gegenständen und bilden unterschiedliche Detailbereiche des Motivs ab. Eine inhaltliche Beschreibung der Suchergebnisse anhand des Beispiels der Notre Dame Kathedrale in Paris erfolgt in Abschnitt 3.1.2.

Weitere mögliche Quellen

Professionelle Agenturfotos Als alternative mögliche Quelle könnten professionelle Agenturfotos in Betracht gezogen werden, um eine hohe Chance auf qualitativ hochwertige Fotos zu bekommen. Je nach Objekt stößt man hier jedoch auf eine sehr begrenzte Verfügbarkeit. Im Gegensatz zu Flickr würde sich der Zugriff zudem komplex und kostspielig gestalten.

Google Street View Export Für viele Gegenden in der Welt wäre ein Export von Fotos aus Google Street View[14] eine vielversprechende denkbare Option, die auch von einigen Arbeiten [2, 82, 71] verwendet wird. Da die Fotos mit einer 360°-Kamera aufgenommen wurden, können bei gleichbleibender GPS-Koordinate mehrere Fotos aus verschiedenen Blickwinkeln extrahiert werden. Mit der Zeitmaschinen-Funktion ist dies für einige Orte zusätzlich aus unterschiedlichen Jahren und Jahreszeiten möglich. Der Nutzen wird hier jedoch auf Objekte begrenzt, die mit dem Auto oder fußläufig erreichbar sind und zudem von Google erfasst wurden. Dadurch sind die Fotos einer Beschränkung auf Straßenebene unterlegen und bieten keine z. B. keine Luftaufnahmen an.

[13] Flickr App Garden: https://www.flickr.com/services/api (Letzter Zugriff: 27.03.2021)
[14] https://www.google.com/intl/de/streetview (Letzter Zugriff: 27.03.2021)

Bestehende Datensätze Für einige Anwendungsfälle wäre der einfachste Weg allerdings, auf bestehende Datensätze ähnlicher Arbeiten zu setzen. Masone und Caputo [40] geben in ihrem Survey-Paper über visuelle Ortserkennung einen chronologischen Überblick über Datensätze, die aus Arbeiten dieser Art seit 2007 bis heute entstanden sind und gruppieren sie zusätzlich nach Skalierung (Welt, Kontinent, Stadt, Gebäude) und Art der annotierten Labels (semantische Informationen, GPS, Pose). Die meisten Datensätze konzentrieren sich jedoch auf Städte-Ebene und streben eine ungefähre Bestimmung der Geoposition auf Basis des erkannten Ortes an oder nutzen semantische Beschreibungen der Orte als Label ohne die Bestimmung von Geokoordinaten.

2.5 Synthetisieren von Datensätzen

Ein Ansatz, der sich grundlegend von den bisher vorgestellten Methoden unterscheidet, ist die synthetische Herstellung von Trainingsdaten, der in diesem Abschnitt behandelt wird.

2.5.1 Überblick

Das wichtigste Kriterium für ein erfolgreiches Training eines Deep Learning Modells ist die Qualität und Menge der Trainingsdaten. Für viele Anwendungsfälle liegen jedoch nur wenige Daten vor oder sie müssen erst aufwendig manuell annotiert werden. Abhilfe können zum einen klassische *Data Augmentation*-Methoden wie Spiegelung, Rotation, Zuschneiden und Transformieren von Bildern schaffen, dessen Wirksamkeit Perez und Wang [50] nachgewiesen haben. Zusätzlich haben sich einige Arbeiten mit der Erstellung von synthetischen Daten beschäftigt, die dann als weitere oder sogar primäre Quelle für das Training verwendet werden können [28]. Die Ergebnisse der klassischen Data Augmentation Verfahren können dann bei der Erstellung synthetischer Bilder direkt mit einbezogen werden [50]. Im Kontext von Trainingsdaten für Faltungsnetze werden für die Synthese Technologien aus der Computergrafik herangezogen. Dafür kommt häufig Software für 3D-Modellierung (bspw. für das Rendern von Gesichtsausdrücken für die Gesichtserkennung [52]) oder *Game Engines* zur Verwendung, mit denen ganze virtuelle Welten modelliert werden, um z. B. die Nutzung von Überwachungskameras [67] oder das autonome Fahren [10] zu verbessern. Eine Vielzahl von Arbeiten mit synthetischen Daten

behandeln die Erkennung von Objektposen und nutzen für das Training 3D-Modelle, die aus multiplen Perspektiven gerendert werden [3, 49].

Tremblay et al. [74] entwickeln mit *DOPE* (Abk. für „*Deep Object Pose Estimation*“) ein System zur Bestimmung der Posen von Haushaltsgegenständen innerhalb eines einzigen Bildes. Diese Arbeit ist im Bereich der Robotik angesiedelt und soll einem Roboter ermöglichen, die trainierten Gegenstände möglichst in Echtzeit erkennen und mit dem Arm greifen zu können. Für das Training erstellen sie einen eigenen Datensatz mit 3D-Modellen einer Auswahl von Haushaltsgegenständen (z.B. Senfflasche, Crackerbox, Brotmesser etc.), den sie als *Falling Things* betitelt haben [73]. Mit der Unreal Engine 4 erstellen sie virtuelle Umgebungen, in die sie die Modelle platzieren und daraus mit einer virtuellen Kamera Bilder aus verschiedenen Blickwinkeln erzeugen und gleichzeitig Kamera- und Objektinformationen exportieren, worunter die Position und Orientierung der Objekte zählen. Diese Informationen können dann beim Training als Label genutzt werden. Als vorbereitende Maßnahme für die vorliegende Masterarbeit wurde auf Basis der Datengenerierung für das DOPE-Projekt ein exemplarischer synthetischer Datensatz mit der Notre Dame Kathedrale in Paris angefertigt [31]. In diesem Zusammenhang wird hier weniger auf die dem DOPE-Projekt zugrunde liegende neuronale Netzarchitektur eingegangen, da es sich bei dieser Arbeit größtenteils um einen anderen Anwendungsfall handelt und nur im Kontext eines Exkurses in Abschnitt 5.4 als Werkzeug verwendet wird. Ferner soll zum besseren Verständnis auf weitere Punkte der Datengenerierung mit der Unreal Engine eingegangen werden, da das Verfahren durch weiterführende Experimente zur Beantwortung der dieser Arbeit zugrunde liegenden Fragestellung weiterhin angewendet wird. In diesem Kontext treten Fragen bezüglich des benötigten Grades des Realismus aus, die im Folgenden kurz näher erläutert werden sollen.

2.5.2 Realismus bei der Datenerzeugung

Bei der Erstellung synthetischer Daten kann der Grad des Realismus sehr unterschiedlich sein. Nach Ferwanda [15] kann zwischen physikalischem Realismus, Fotorealismus und funktionalem Realismus unterschieden werden, wobei ersterer hier keine Anwendung findet und deshalb nicht näher erläutert wird. Mit fotorealistischen Bildern soll die virtuelle Umgebung die gleiche visuelle Reaktion des Auges wie die Realität auslösen. Das erzeugte Bild soll demnach nicht von einer Fotografie unterscheidbar sein. Der funktionale Realismus hingegen besagt, dass ein Bild die gleichen visuellen *Informationen* wie die Realität enthalten muss, mit denen bestimmte Aufgaben ebenso erfüllt werden können.

Dabei kann es sich bspw. um gezeichnete Montageinstruktionen handeln oder Computerspiele, die auch ohne fotorealistische Szenen realistische Interaktionen ermöglichen. Die funktionalen Informationen wären dann Form, Größe, Position etc. der abgebildeten Objekte.

Realitätslücke

Als Realitätslücke (engl. *Reality Gap*) wird die Diskrepanz zwischen den Eigenschaften simulierter Umgebungen und der Realität beschrieben und ist besonders in der Robotik eine große Herausforderung, da sich das Verhalten eines Roboter dadurch zu nah an der Simulation bewegen kann [70]. In der Arbeit von Tremblay et al. [72] wirkt sich diese Diskrepanz insofern aus, als dass sich das Training des neuronalen Netzes zu sehr an die Eigenschaften der synthetischen Daten anpasst und folglich schlecht für allgemeine Situationen generalisieren kann. Zu berücksichtigende Faktoren bei der Erstellung virtueller Umgebungen sind besonders der Verdeckungsgrad der zu trainierenden Objekte, verschiedene Licht- und Schattenverhältnisse bei unterschiedlichen Posen und die Beschaffenheit der Texturen.

Domain Randomization

Um die Realitätslücke möglichst gut zu schließen und das neuronale Netz robuster gegenüber Störfaktoren zu machen, greifen Tremblay et al. [72] auf die von Tobin et al. [70] eingeführte *Domain Randomization* zurück. Dabei handelt es sich um ein sehr einfaches und effizientes Verfahren, mit dem durch die zufällige Anzahl und Anordnung von störenden bzw. verdeckenden Objekten und variabler Lichtverhältnisse, wechselnder Texturen und Hintergründen verschiedene Variationen der virtuellen Umgebung erstellt werden. Durch die Vielfältigkeit soll die ursprüngliche virtuelle Umgebung vom neuronalen Netz ebenso nur noch als eine mögliche Variation wahrgenommen und letztendlich dazu gezwungen werden, sich auf die wesentlichen Merkmale, also die zu trainierenden Objekte, zu konzentrieren.

NVIDIA Deep Learning Dataset Synthesizer (NDDS)

Für die Erstellung des synthetischen Datensatzes haben die Autoren den *NVIDIA Deep Learning Dataset Synthesizer (NDDS)* [69], ein Plugin für die Unreal Engine, ent-

wickelt. Mit dem Plugin können beliebig viele zu trainierende Objekte in die Unreal 3D-Umgebung eingesetzt werden. Über verschiedene Parameter können Farben, Texturen und Position und Orientierung der Objekte bei wechselnden Lichtverhältnissen zufällig verändert werden. Zudem bietet das Plugin die Möglichkeit, beliebig viele Störobjekte – sogenannte Distraktoren – ebenso mit zufälligem Erscheinungsbild und Pose in der Umgebung zu platzieren. Über eine Exportfunktion werden während der Simulation Screenshots der Szene erstellt und als **.png*-Datei nebst je einer Metadaten-Datei pro Bild gespeichert. In den Metadaten befinden sich Position und Orientierung der Kamera und der zu trainierenden Objekte, die dann für das anschließende Training genutzt werden können. Für dieses Verfahren ist es unerheblich, ob fotorealistische oder funktionale Informationen generiert werden sollen. Wird in der Engine keine fotorealistische Umgebung modelliert, kann das Plugin z.B. durch das Einsetzen zufälliger Hintergründe willkürliche Varianten der virtuellen Umgebung im Sinne der Domain Randomization erzeugen. Für weitere Details bezüglich des Einsatzes des NDDS-Plugins sei an dieser Stelle auf [31] verwiesen. Im Kapitel „Datenmaterial“ (Kapitel 3) wird sich im Zuge der Datengenerierung für die anschließenden Experimente jedoch auch noch weiter mit Details des Plugins beschäftigt.

2.6 Analyse Zusammenfassung

In der Analyse wurden die theoretischen Grundlagen, die für die Rekonstruktion von Geopositionen eines einzelnen Bildes nötig sind, ausgearbeitet und dadurch ein Einblick in die Teilbereiche dieser Arbeit gegeben. Thematisch lehnt sich diese Arbeit an Arbeiten der Bereiche der visuellen Ortserkennung und visuell-basierten Lokalisierung an, sodass mit der Vorstellung von Feature-Detektoren und -Deskriptoren eine Einführung in die Thematik der Bilderkennung samt Anwendungsbeispielen anhand analytischer Methoden aus dem Bereich der Computer Vision erfolgt ist. Auch wenn sich die analytischen Methoden bereits gut für die Objekterkennung eignen, sind sie je nach Anwendungsfall anfällig gegenüber Verdeckung und Transformation von Objekten und aufwändig in der Realisierung.

Als weiteres Forschungsgebiet wurde deshalb das maschinelle Lernen mit Fokus auf die Verwendung von tiefen neuronalen Netzen vorgestellt. Da sich in dem Bereich der Bilderkennung bzw. -verarbeitung die Convolutional Neural Networks als „state-of-the-art“-Lösung zuerst für Klassifikationsprobleme etabliert haben, erfolgte eine genauere Ana-

lyse dieser Faltungsnetze mit Verweisen auf gängige darauf aufbauende Architekturen insbesondere der InceptionV3-Architektur. Als Quintessenz konnte herausgearbeitet werden, dass die Verwendung gängiger CNN-Architekturen die aufwändige Erstellung von Feature-Detektoren und -Deskriptoren ersetzt. Diese lassen sich zudem generisch für eine Vielzahl verschiedener Anwendungsfälle, besonders bei großen Datenmengen durch deutlich gesteigerte Performance verglichen mit den analytischen Ansätzen, einsetzen.

Bezogen auf die Positionsbestimmung der Kamera eines einzelnen Bildes wurden in Abschnitt 2.3 Arbeiten herangezogen, die diese Problemstellung als Klassifikationsproblem definieren und mit Arbeiten auf analytischer Basis verglichen. Die Analyse hat ergeben, dass die Bestimmung von GPS-Koordinaten mit einer Klassifikation funktioniert, die Genauigkeit in den vorgestellten Arbeiten allerdings erst ab einem Umkreis von einem Kilometer bestimmt wird. An dieser Stelle stellt sich folglich die Frage, ob eine genauere Bestimmung der GPS-Koordinaten mit einer Klassifikation möglich ist. Im Rahmen der Aufgabenstellung dieser Arbeit soll dies nachfolgend überprüft und deshalb ein Training für ein bekanntes Objekt durchgeführt werden, das als Referenz für die Rekonstruktion der Geoposition dienen soll.

Für die Erzeugung des Datensatzes wurden in Abschnitt 2.4 deshalb Bedingungen an den Datensatz gestellt wie Größe, Verfügbarkeit und Qualität der Daten. Da für den Datensatz GPS-Koordinaten zum Trainieren des Netzes benötigt werden, können viele potenzielle Quellen für Internetfotos wie klassische Suchmaschinen nicht verwendet werden. Die Online-Fotoplattform Flickr bietet allerdings für viele Fotos Metadaten und eine API zur programmatischen Abfrage aller Daten an, sodass sie sich grundsätzlich als gute Quelle anbietet, wenngleich die Qualität der Ergebnisse noch viel Spielraum nach oben lässt.

Deshalb wurde als alternative Quelle mit dem NVIDIA Deep Learning Dataset Synthesizer [69] ein Ansatz zur synthetischen Herstellung von Trainingsdaten vorgestellt und in dem Zusammenhang die Methode der Domain Randomization beschrieben – ein Verfahren mit dem ohne großen Rechenaufwand tausende Fotos eines Objektes in zufälliger Erscheinung und Umgebung im Sinne des funktionalen Realismus generiert werden können. Einhergehend fiel der Begriff der Realitätslücke, die sich bei der Objekterkennung zwischen synthetischen Daten und der realen Welt aufspannt. Diese sollte möglichst gering für eine erfolgreiche Anwendung dieses Verfahrens auf die Aufgabenstellung ausfallen.

Basierend auf den Erkenntnissen der Analyse wird in den folgenden Kapiteln dieser Arbeit die Frage behandelt, mit welcher Genauigkeit die Kameraposition aus einem einzigen

Foto rekonstruiert werden kann. Als Trainingsmodell soll ein Klassifikator auf Basis der InceptionV3-Architektur implementiert und konträr zu den vorgestellten Arbeiten ein Trainingsdatensatz auf einer kleinen Skala im Bereich des Umkreises einer bekannten Sehenswürdigkeit erstellt werden. Dafür werden sowohl Flickr als reale Datenquelle als auch mit dem NDDS hergestellte synthetische Daten für das Training herangezogen.

3 Datenmaterial

Da die Erstellung der Datensätze für das Training des neuronalen Netzes einen entscheidenden Teil dieser Arbeit darstellt, werden in diesem Kapitel sowohl die Beschaffung der realen Fotos als auch die Erstellung synthetischer Fotos beschrieben. Beiden Datensätzen liegt die gleiche Vorgehensweise bei der Erstellung der Labels zugrunde, welche sich jedoch unterschiedlich auswirkt, sodass auf diese in beiden Abschnitten eingegangen wird.

3.1 Realer Datensatz

3.1.1 Datenbeschaffung

Aus der Analyse ist hervorgegangen, dass sich für die Beschaffung großer Datenmengen besonders Crowd-basierte Fotoplattformen im Internet anbieten können. Die Wahl fiel deshalb auf Flickr, da dort der Fokus auf semi-professionelle Fotos gelegt wird, die in vielen Fällen Geopositionen in Form von GPS-Tags enthalten. Da diese weniger komplex sind als z. B. intrinsische und extrinsische Kameraparameter wie sie in [61] genutzt wurden, sollen sie als Label für das Training dienen. Die Flickr API bietet Zugriff auf die öffentlich verfügbaren Fotos und stellt mehrere Filterkriterien zur Verfügung. In einem dieser Arbeit vorangegangenen Projekt [30] wurde ein Download-Client geschrieben, der mit einer für Python adaptierten FlickrAPI Bibliothek[15] realisiert wurde. Als Filterkriterium konnte die GPS-Position der Fotos genutzt werden, sodass nur Fotos mit einem GPS-Tag geladen wurden, die sich im Einzugsgebiet von etwa 5 km Radius um die Notre Dame befinden. Die meisten hochgeladenen Fotos wurden mit einem oder mehreren Stichworten markiert, sodass Filterregeln mit Suchbegriff-Kombinationen aufgestellt wurden, um möglichst relevante Fotos zu erhalten, die als Motiv das gewünschte Gebäude enthalten

[15] Python Implementierung der Flickr API: https://pypi.org/project/flickrapi (Letzter Zugriff: 07.04.2021)

(z. B. Wortvariationen aus „Notre Dame“, „Notredame“ etc.). Fotos mit häufig auftretenden Stichworte, die auf Nahaufnahmen anderer Motive hindeuten wie bspw. „Café“ oder „Flowers“, wurden explizit ausgeschlossen. Die API bietet einen weiteren Parameter zum Filtern von Innenaufnahmen an. Die Nutzung dieser für diesen Zweck sehr nützlichen Funktion hatte jedoch leider keine Wirkung auf die Ergebnisse, dessen Auswirkungen auf den Datensatz im Folgenden beschrieben werden.

3.1.2 Analyse

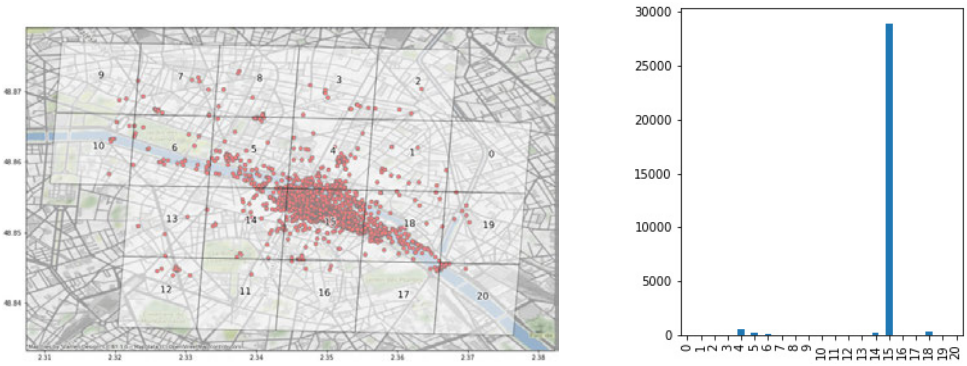
Größe des Datensatzes

Mit dem soeben beschriebenen Download-Verfahren konnte eine Gesamtanzahl von 35.362 Fotos mit einer Größe von 7,9 GB heruntergeladen werden. Anschließende Dateiüberprüfung, Auflösungsanpassung auf 256 x 256 Pixel, Zuordnung der separat gespeicherten Metadaten und das Entfernen von Duplikaten reduzierte die Zahl letztendlich auf 30.437 nutzbare Fotos bei einer Größe von 1,8 GB.

Verteilung

Abbildung 3.1 zeigt die Verteilung der Fotos anhand der Geo-Koordinaten auf einem Kartenausschnitt im Bereich der Notre Dame. Für eine grobe Analyse und einer besseren Beschreibung eignet sich hier schon die Einteilung der Fotos in ein Raster, das erst im nächsten Abschnitt näher erläutert wird. Für die Einteilung wurde die S2 Geometry Library[16] mit dem Raster Level 13 verwendet. Laut Dokumentation entspricht eine Zelle dann (je nach Region) durchschnittlich 1,27 km² Fläche. In diesem Fall ist eine Zelle schließlich ca. 830 m breit und etwa 1,2 km hoch. Zu sehen ist in Abbildung 3.1a eine erwartbare Anhäufung von Fotos besonders innerhalb der Zelle 15 und in deren näherem Umkreis. Hier befindet sich die *Île de la Cité*, die Stadtinsel in Paris, auf der sich am östlichen Ende die Notre Dame Kathedrale befindet. Durch die Insellage ist hier eine gute Abdeckung von der Südseite des Gebäudes gegeben, da touristische Fotos von Fähren aus aufgenommen werden können. Zudem ist das gesamte Areal rund um die Notre Dame fußläufig begehbar, wodurch eine Abdeckung aller Seiten der Kathedrale gegeben ist. Das Balkendiagramm in Abbildung 3.1b zeigt die Anzahl vorhandener Fotos pro Zelle. 28.874 Fotos befinden sich in diesem Areal, was 95 % aller Fotos des Datensatzes entspricht. Bis

[16] S2 Geometry Library: https://s2geometry.io/resources/s2cell_statistics (Letzter Zugriff: 15.01.2021)



(a) Geoverteilung der aus Flickr gewonnenen Daten

(b) Anzahl der Fotos pro Klasse bei *S2 Level 13*

Abbildung 3.1: Verteilung der 35.000 aus Flickr gewonnenen Fotos mit einer Zellengröße von durchschnittlich 1,27 km²

auf die Zellen 4 bis 6, 14 und 18, die mehr als 50 Samples enthalten, treten sonst nur vereinzelte Vertreter innerhalb des ca. 20 km² großen Areals auf.

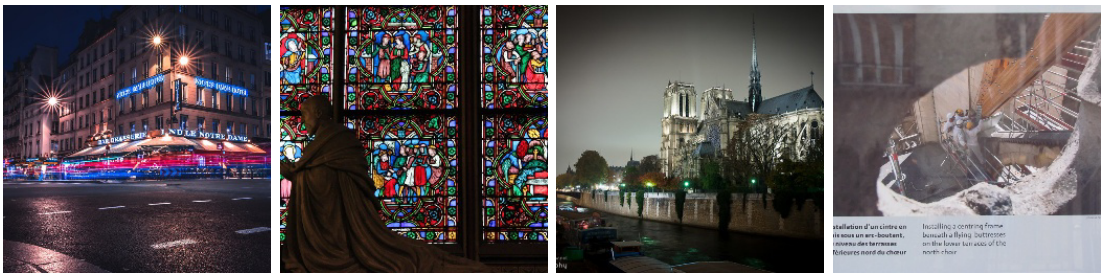
Inhalt

Hays und Efros haben in ihrer Arbeit IM2GPS [23] bereits 2008 festgestellt, dass es äußerst schwierig ist, einen hochqualitativen Datensatz aus Daten zu erstellen, die von Nutzer:innen erstellt und auf öffentlichen Plattformen im Internet hochgeladen wurden, da sie z. B. dazu tendieren, jedes beliebige Foto unabhängig vom Inhalt mit GPS-Tags zu versehen. Deshalb liefert nur eine Kombination aus GPS-Tag Filterung und ortsbeschreibende Stichworte brauchbare Ergebnisse. Im Falle der Notre Dame wurde in der vorausgehenden Arbeit [30] bereits gezeigt, dass trotz der Filterkombinatorik der Inhalt der Fotos des Datensatzes äußerst divers ist. Viele Fotos enthalten erwartungsgemäß keine freie Sicht auf die Kathedrale, da sie durch Bäume, Menschen (Portraits, Gruppenfotos, Passanten) oder Fahrzeuge verdeckt werden. Ferner finden sich jedoch auch Motive von Pflanzen und Cafés ohne jeglichen Bezug zur Geoposition und dem Objekt von Interesse wieder. Charakteristisch für die Notre Dame sind das Hauptportal in der Mitte der Westfassade mit ihren zahlreichen Figuren und der Grotesken auf dem Dach des Gebäudes, von denen es jeweils eine Reihe von Fotos gibt. Hier stellt sich die Frage, inwieweit diese Fotos das Training beeinflussen bzw. ob das trainierte Netz in der Lage ist, die Geoposition anhand dieser Merkmale zu lokalisieren. Was sich zudem als proble-

matisch erweisen kann, sind zahlreiche Innenaufnahmen trotz des aktiven Filters in der Flickr API mit Makroaufnahmen von Kerzen und Dekorationen.

(a) Zelle 0

(b) Zelle 11



(c) Zelle 15

(d) Zelle 20

Abbildung 3.2: Auswahl fehlerhafter Daten im realen Datensatz

Zuletzt sei hier auch auf die Ungenauigkeiten bzw. Inkonsistenz der vorhandenen GPS-Tags eingegangen. Ein Beispiel zeigt Abbildung 3.2a mit Fotos aus Zelle 0 der Abbildung 3.1a), deren Motive (wenn auch im zweiten Fall stark verfremdet) die Westfassade zeigen, jedoch geografisch weit im Osten angesiedelt werden. Dieses Verhalten ist bei mehreren Fotos im weiten Umkreis zu verzeichnen, besonders häufig tritt eine fehlerhafte Zuordnung allerdings für Koordinaten auf, die sich direkt auf dem Gebäude befinden. Es kann davon ausgegangen werden, dass viele Fotos mit Standardwerten für Längen- und Breitengrad der Notre Dame Kathedrale versehen wurden – unabhängig davon, welche Motive sie zeigen. Eine manuelle Überprüfung aller Fotos wäre sehr mühsam und

nicht zielführend, automatisiertes Überprüfen und Herausfiltern der fehlerhaften Daten soll laut Zieldefinition dieser Arbeit (siehe Abschnitt 2.6) jedoch mithilfe dieser Daten erst erzeugt werden. Somit folgen im Kapitel 4 Experimente, die die Nutzbarkeit dieser Daten anhand eines exemplarischen Trainings zeigen.

3.1.3 Erstellung der Label

Als Label werden wie in Abschnitt 2.3 beschrieben nicht die Geo-Koordinaten der Fotos genutzt, sondern mithilfe der S2 Geometry Bibliothek einzelnen Sektoren, die im Folgenden *S2-Zellen* genannt werden, zugeordnet. Die S2-Zellen bestehen aus einer eindeutigen 64-Bit langen ID, die aus Gründen der Lesbarkeit und Visualisierung mit einem durchnummerierten Index versehen werden. Dieser *S2-Label Index* dient dann wiederum als Label und entspricht einer Klasse für den Klassifikator.

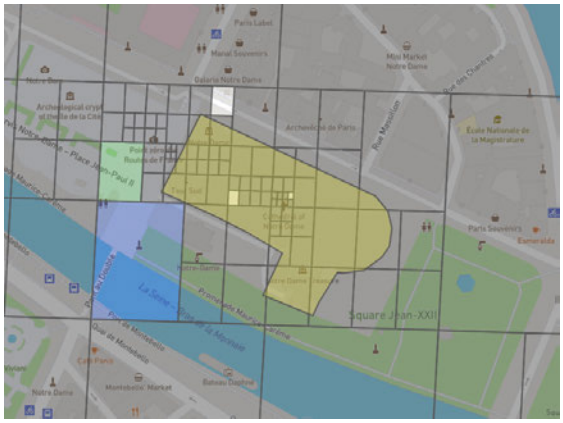
Die Analyse der Verteilung der Fotos hat gezeigt, dass sich die Mehrheit auf einen kleinen Bereich rund um die Notre Dame herum beschränkt. Ein sinnvolles Training mit einem Klassifikator ist nur mit einer einigermaßen ausgeglichenen Aufteilung der Trainingsdaten möglich. Es muss also eine Zellengröße gefunden werden, in die sich die Menge bestmöglich verteilen kann. Um die Aufteilung in Abbildung 3.1b in kleinere Klassen einzuteilen, wurde in [30] neben statischen Varianten auch ein dynamisches Verfahren wie in [78] entwickelt, mit dem sich die Größe der Zellen dynamisch anhand der Menge der darin enthaltenen Fotos anpasst. So kommen zwei Verfahren zum Einsatz:

Statische Zelleinteilung:

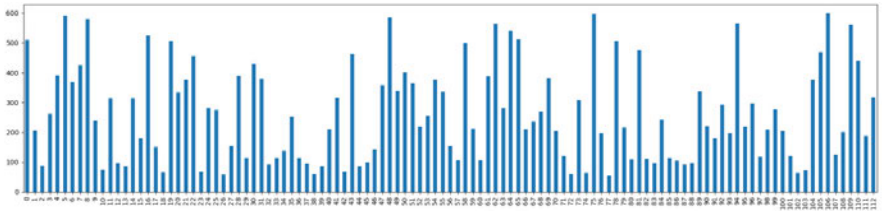
1. Bestimmung der Minimal- und Maximalanzahl von Fotos pro Zelle,
2. Entfernung von Zellen mit einer zu geringen Anzahl von Fotos ($<$ Minimum),
3. Entfernung überschüssiger Fotos aus Zellen mit zu vielen Fotos ($>$ Maximum).

Dynamische Zelleinteilung:

1. Bestimmung der Minimal- und Maximalanzahl von Fotos pro Zelle,
2. Entfernung von Zellen mit einer zu geringen Anzahl von Fotos ($<$ Minimum),
3. rekursives Vierteln von Zellen mit überschüssigen Fotos, bis keine Zelle mehr das Maximum überschreitet.



(a) Kartenausschnitt mit dynamischem Raster. *Markierungen:* Umriss der Notre Dame (gelb), Level 17-Zelle (blau), Level 18-Zelle (grün)



(b) Verteilung der Fotos auf dynamische Klassen

Abbildung 3.3: Dynamische Zelleneinteilung
mit *S2-Minimum:* 50, *S2-Maximum:* 600. *Start-S2-Level:* 13

Abbildung 3.3a zeigt einen nahen Kartenausschnitt mit der Notre Dame (gelbe Markierung) im Zentrum und einem darübergelegten dynamischen Raster. Als Minimalwert wurden 50 und als Maximalwert 600 Fotos pro Zelle eingestellt mit dem S2-Level 13 als Startwert für die Rekursion. Die blau markierte Zelle entspricht einem S2-Level 17 mit einer Zellengröße von 3600 m^2, während die grüne Markierung einer S2-Zelle des Levels 18 mit 900 m^2 Größe entspricht. Hier ist nun gut zu erkennen, wie weit sich die Zelle 13 aus der groben Einteilung in Abbildung 3.1a mit diesem Verfahren aufteilen lässt. Jedoch offenbart sich hier auch wieder das Problem der ungleichen Verteilung und des Inhalts der Daten: Die Rekursion erfolgt nach diesem Muster bis hin zum Level 21. Die kleinsten Zellen befinden sich vor der Westfassade, da dort der vermutlich beliebteste Spot für touristische Fotos lokalisiert ist. Die meisten Zellen jedoch sind innerhalb oder auf dem Gebäude angesiedelt. Das lässt die Schlussfolgerung zu, dass es sich hierbei um die in Abschnitt 3.1.2 beschriebenen Innenaufnahmen und falsch getaggten Fotos handelt.

Die rekursive Vorgehensweise wirkt sich natürlich auf die Anzahl der zu trainierenden Klassen und die Granularität der Rekonstruktion der Geo-Positionen aus. Aus einem Datensatz können folglich durch unterschiedliche Einteilung der Labels mehrere Datensatz-Varianten gebildet werden. Tabelle 3.1 zeigt die Auswirkung dieser Variationen auf die Anzahl der Klassen und der jeweils zugeordneten Anzahl an Fotos. Die fett gedruckten Zeilen entsprechen jeweils einer Rastereinteilung ohne Definition von Minimal- und Maximalwert. Bei Nutzung des S2-Level 18 (siehe Datensatz-Variantion *Real18*) wären dies 910 Klassen, da die Mehrheit der Klassen nur jeweils ein Sample beinhalten würden. Diese Varianten wurden nicht trainiert, stattdessen erfolgen Experimente mit Minimum- und Maximumkriterien wie in Zeile *Real18.800* dargestellt, die die Anzahl der zu trainierenden Klassen auf nur noch 65 senkt.

				Erlaubte Fotos pro Klasse		**Vorhandene Fotos pro Klasse**				
Datensatz-Variante	**S2-Level**	**Zellengröße (in m²)**	**Anz. Klassen**	**Min.**	**Max.**	**Min.**	**Max.**	**Durch-schnitt**	**Median**	**Gesamt**
Real13	**13**	**921107**	**21**	**-**	**-**	**2**	**28876**	**1449**	**15**	**30437**
Real13.600dyn	13-21	variabel	113	50	600	54	599	263	220	29720
Real17	**17**	**3599**	**485**	**-**	**-**	**1**	**7742**	**63**	**2**	**30437**
Real17.800			44	50	800	51	800	269	143	11843
Real17.600			44	50	600	51	600	237	143	10443
Real18	**18**	**899**	**910**	**-**	**-**	**1**	**3496**	**33**	**2**	**30437**
Real18.600			65	50	600	50	600	219	141	14251
Real21	**21**	**14**	**5590**	**-**	**-**	**1**	**304**	**5**	**1**	**30437**

Tabelle 3.1: Datensatz-Varianten durch Anpassung der Rastergröße

3.1.4 Bewertung

Wie die manuelle Stichprobe des Inhalts der gewonnenen Daten ergeben hat, sind diese extrem verrauscht und eine relativ gleichmäßige Verteilung der Daten auf die Klassen ist nur mit größeren Anpassungen der Klassengrößen möglich. Dadurch, dass einige Fotos offensichtlich falsch getaggt wurden, sind keine Ground Truth Daten, also Daten, die die korrekten realistischen Gegebenheiten repräsentieren, vorhanden. Vorläufige Testansätze in [30] mit minimalistischer Deep Learning Architektur deuteten bereits auf unzufriedenstellende Ergebnisse und geringe Brauchbarkeit dieses Datensatzes hin. Deshalb erfolgte in einer weiteren Arbeit [31] die Erstellung eines synthetischen Datensatzes als weiterer Ansatz.

3.2 Synthetische Datensätze

Im vorherigen Abschnitt wurde deutlich, dass die Erstellung eines geeigneten Datensatzes aus realen frei zugänglichen Fotos mit den dazugehörigen Labels ein komplexes Unterfangen mit geringer „Ausbeute“ ist. Bedingt durch die begrenzte Verfügbarkeit von Fotos wurde u. a. bei der Erstellung der Labels auf ein dynamisches Zelleneinteilungsverfahren gesetzt. Ziel der Erstellung synthetischer Trainingsdaten ist es, unabhängig von diesen Beschränkungen zu sein, da bei der Generierung der Daten über die Anzahl und Geoposition der Fotos frei entschieden werden kann. Es wird also davon ausgegangen, dass die generierten Fotos in ein statisches Raster wie soeben beschrieben eingeteilt werden können. Dieser Abschnitt beschäftigt sich deshalb primär mit dem Erstellungsvorgang der synthetischen Bilder.

3.2.1 Wahl der Methodik

Die Analyse hat ergeben, dass das NDDS-Plugin [69] für die Unreal Engine ein vielversprechendes Tool für die Erstellung von synthetischen Fotos sein kann. Tremblay et al. [72] konnten zeigen, dass durch die Domain Randomization-Methode eine effiziente Möglichkeit existiert, die Realitätslücke zwischen realen und synthetischen Daten möglichst zu schließen. In seiner Masterarbeit hat Dustin Spallek [63] einen Datensatz nach diesem Vorbild mit sowohl fotorealistischen als auch domänenrandomisierten Fotos erstellt und drei Haushaltsgegenstände trainiert. Die Genauigkeit des darauf trainierten Deep Learning Netzwerks wurde in einem Versuchsaufbau gemessen und validiert. Hoher Anspruch an die Qualität geht jedoch stets auf Kosten der Berechnungsdauer (siehe Abschnitt 2.5): aufgrund einer Trainingsdauer von etwa 11 Tagen auf der Hochleistungs-Renderfarm der HAW Hamburg [44] ist die Praxistauglichkeit für dieses Vorgehen dadurch eingeschränkt. Zumindest für Objekte, die um ihre sechs Freiheitsgrade (6-DoF-Objekte[17]) rotiert werden können, konnte allerdings bereits bei der Verwendung von ausschließlich domänenrandomisierten Fotos eine vielversprechende Erkennungsrate erreicht werden, was durch die Messungen bestätigt wurde. Durch die Nutzung dieser Vorgehensweise wird ein geringer Aufwand bei Modellierung und Training ermöglicht. Im Zuge dieser Arbeit soll deshalb ausschließlich ein domänenrandomisierter Datensatz für das Training der Notre Dame Kathedrale erzeugt und evaluiert werden, inwiefern sich dieses Vorgehen für große

[17] 6-DoF (6 Degrees of Freedom): Bewegungsfreiheit eines starren Körpers im dreidimensionalen Raum anhand der drei lotrechten Achsen [48]

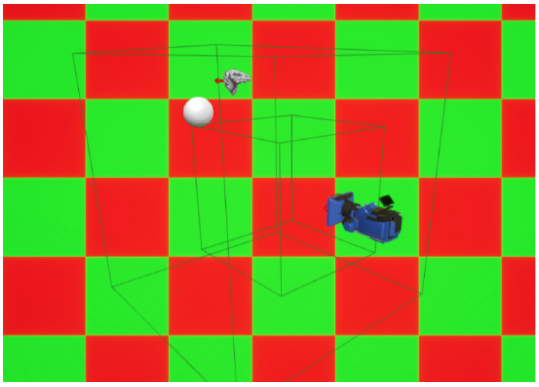
Objekte wie Sehenswürdigkeiten im Gegensatz zu Haushaltsgegenständen eignet. Für diesen neuen Anwendungsfall wurden einige Anpassungen an der Vorgehensweise für die Synthese der Foto vorgenommen, die im Folgenden zusammengefasst werden.

3.2.2 Szenen-Aufbau

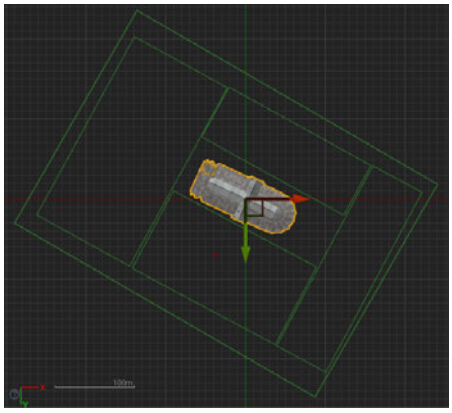
Ursprünglicher Szenen-Aufbau für 6-DoF Objekte Da der NVIDIA Deep Learning Dataset Synthesizer [69] primär für das Training von Haushaltsgegenständen entwickelt wurde, um das Greifen durch Roboterarme zu ermöglichen, zielt der Standardaufbau der Simulation darauf ab, die sechs Freiheitsgrade und damit die Pose eines Objektes im freien Raum zu trainieren. Um dies zu realisieren, werden 3D-Modelle der zu trainierenden Objekte in die Unreal Engine importiert. Mit dem Plugin können anschließend sogenannte dreidimensionale *Trigger Volumes* erstellt werden, die einen Bereich definieren, in dem die zu trainierenden Objekte während des Exportvorgangs zufällig erscheinen und sich bewegen und rotieren. Die Kamera wird statisch positioniert und zeichnet die Aktivitäten innerhalb der Trigger Volumes auf (Abbildung 3.4a). Das äußere große Trigger Volume in Abbildung 3.4a definiert den Bereich für die Distraktoren, während der gekachelte Hintergrund dem Bereich entspricht, der während der Generierung durch zufällige Hintergrundbilder aus dem COCO-Datensatz [36] ersetzt wird. Die Objektgrößen und Abstände zur Kamera werden in der Unreal-eigenen *Unreal Unit (UU)* angegeben, die in der Standardeinstellung einem Zentimeter entspricht. Damit die Posen später richtig bestimmt werden können, entsprechen alle Angaben den realen Größen.

Angepasster Szenen-Aufbau Für das Training bekannter räumlich statischer Objekte wie Sehenswürdigkeiten kann das Verfahren in dieser Form nicht verwendet werden, da die Objekte nicht um alle Rotationsachsen im Sinne der sechs Freiheitsgrade gedreht werden können. Es wäre schließlich wenig sinnvoll, den Boden der Notre Dame zu modellieren und aufzuzeichnen, wenn dieser in der Realität gar nicht existiert und auch dementsprechend im realen Datensatz gar nicht auftauchen könnte. Statt die zu trainierenden Objekte dynamisch agieren zu lassen, war die Überlegung, das Objekt stattdessen statisch in der Szenerie zu platzieren und die Kamera dynamisch um das Objekt herumzubewegen. Somit wurde ein 3D-Modell der Notre Dame auf Turbosquid[18]

[18] Webportal für den Vertrieb von 3D-Modellen für Computergrafik: https://turbosquid.com (Letzter Zugriff: 14.02.2021)



(a) Statische Aufnahmestelle mit dynamisch agierenden Objekten



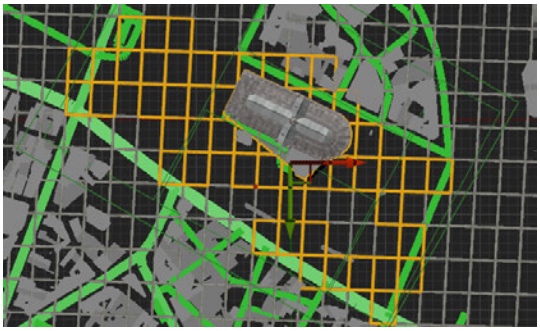
(b) Vier Aufnahmestellen für dynamische Kameras mit statischem Objekt

Abbildung 3.4: Angepasster Simulationsaufbau für die *Domain Randomization*-Methode

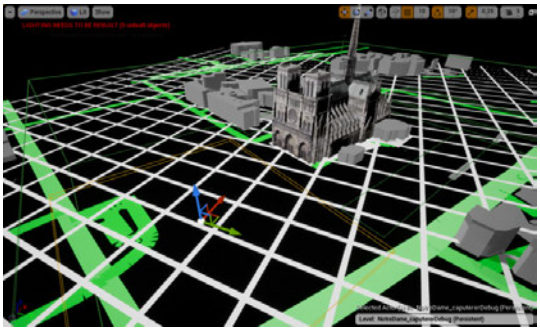
erworben und auf die realen Maße skaliert in die Unreal Szene importiert. Für die genaue Bestimmung von Position und Rotation des Modells wurde ein Kartenauszug der Umgebung aus OpenStreetMap[19] mithilfe eines entsprechenden Plugins in das Unreal-Koordinatensystem importiert und das Objekt daran ausgerichtet. Für die Kamera wurden mehrere Trigger Volumes (im Folgenden *Camera Volumes* genannt) erstellt, in denen sie sich frei bewegen kann, sodass Aufnahmen von jeder Seite der Kathedrale erstellt werden können (Abbildung 3.4b). Die Kamerahöhe wurde mittels z-Achse auf etwa 1,5 m fixiert, um die Aufnahmeposition von Touristen zu simulieren und unrealistische Aufnahmeorte zu verhindern. Um die Blickrichtung steuern zu können, wurden Fixpunkte am tiefsten und höchsten Punkt in der Mitte des 3D-Modells angebracht, zwischen denen die Kamera unabhängig von ihrer Position zufällig wechselt. Mit dem OpenStreetMap-Plugin ließ sich weiterhin das S2-Raster in gewünschter Granularität visuell in die Szene integrieren, sodass beim Erstellen der Kamera-Volumes die Anzahl der dadurch abgedeckten Zellen bzw. Klassen für das anschließende Training eingesehen werden konnten (siehe Abbildung 3.5). Eine alternative Kameraführung könnte auch mit einer orbitalen Umkreisung des Objektes implementiert werden. Dadurch würde allerdings eine Perspektive eingenommen werden, die mit Drohnen- oder Flugzeugaufnahmen vergleichbar wäre, weshalb sie hier keine Verwendung findet. Der Einsatz von Distraktoren wurde wie im ursprünglichen Simulationsaufbau mit einem all umschließenden Trigger Volume realisiert.

[19] https://www.openstreetmap.de/ (Letzter Zugriff: 14.02.2021)

Damit ist gewährleistet, dass unabhängig von der Aufnahmestelle Distraktoren das Bild stören können. Eine genauere Beschreibung des Simulationsaufbaus findet sich in [31].



(a) Draufsicht mit S2-Zellen Raster des realen Datensatzes (gelbe Markierung)



(b) Perspektivische Ansicht

Abbildung 3.5: Notre Dame Szene mit eingeblendeter OpenStreetMap Umgebung und S2-Zellen-Raster in UE4

3.2.3 Annotation der Objekte

Während des Exportvorgangs werden im ursprünglichen Aufbau die zu trainierenden Objekte zwecks der Posenbestimmung mit Begrenzungsrahmen annotiert. Diese sind zwar nicht in den exportierten Bildern zu sehen, die Positionen der Kanten werden jedoch in der exportierten Metadaten-Datei angegeben und können anschließend für das Training des neuronalen Netzes als Label genutzt werden. Dadurch, dass das Modell der Notre Dame in diesem Versuchsaufbau statisch platziert wurde, sind die Poseninformationen konstant und müssen nicht exportiert werden. Stattdessen wird die 3D-Position der Kamera gespeichert. Mit dem Mittelpunkt des Notre Dame Modells als Referenzwert lässt sich der Abstand der Kamera zum Objekt messen. Um Metadaten analog zu den realen Daten aus Abschnitt 3.1.3 zu erzeugen, wird die Geoposition aus Längen- und Breitengrad benötigt. Da dafür die Höheninformation der Kamera nicht relevant ist, werden dessen x- und y-Koordinaten im Unreal Koordinatensystem für die Umrechnung verwendet und mithilfe des UTM-Koordinatensystems (siehe [31]) in GPS-Koordinaten umgerechnet.

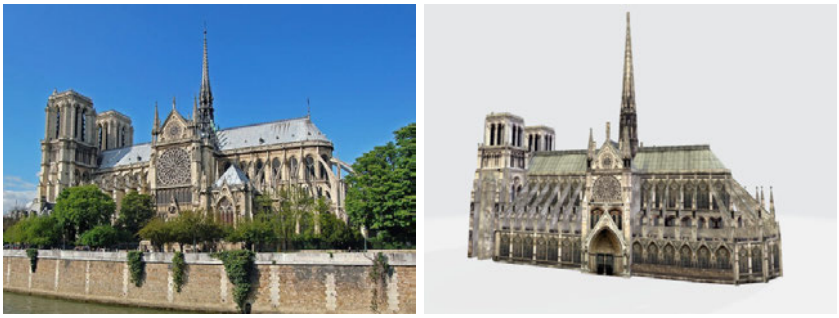
3.2.4 Bildinhalt

Im Gegensatz zur Verwendung realer Fotos als Quellmaterial kann bei synthetischen Daten vor der Erzeugung bestimmt werden, welcher Inhalt auf den Fotos zu sehen sein

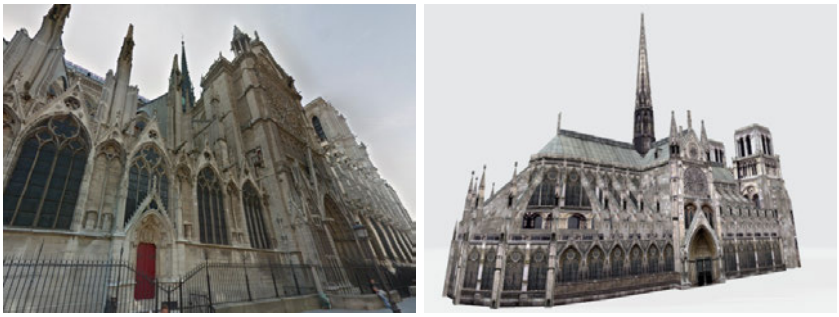
soll. In [31] wurde das soeben beschriebene Verfahren genutzt, um initiale Datensätze zu erstellen und mit dem realen Datensatz zu vergleichen. Wie später im Experimente-Teil in Kapitel 4 beschrieben wird, ist u. a. die Komplexität des Datensatzes ausschlaggebend für den Trainingserfolg, dass einige Punkte auf inhaltlicher Basis hier nun kurz erläutert werden.

3D-Modell Bei der Wahl des 3D-Modells wurde versucht, einen Kompromiss aus Detailgrad bzw. Realismus und finanzieller Erschwinglichkeit zu finden, wobei die Preisspanne je nach Qualität des Modells sehr groß sein kann. Somit wurde ein Modell mit wenigen Polygonen, aber größtenteils realistischer Textur verwendet, auch in Hinblick darauf, dass das hier gewählte synthetische Verfahren ebenfalls für weitere Modelle genutzt werden können soll. Demzufolge wäre interessant zu erfahren, inwieweit Low-Poly Modelle ausreichend zur Erreichung des Ziels sein können.

Ein Vergleich des 3D-Modells mit dem realen Gebäude (Abbildung 3.6) zeigt allerdings einige Unterschiede: Zum einen fehlt die kleine Kapelle an der Südfassade vollständig und die Modellierung des Seiteneingangs gleicht die der Nordfassade (vgl. Abb. 3.6a und 3.6b). Weiter fehlt die kleine Eingangstür auf der Nordseite und die Modellierung unterscheidet sich zur Realität in Bezug auf die Höhe der beiden Haupttürme und die Fenster auf der Ostseite bzw. Rückseite (vgl. Abb. 3.6c). Allgemein finden sich im 3D-Modell weniger Details was Art und Anzahl der Figuren, Formen und Muster auf der Fassade anbelangt und insgesamt ist die Auflösung der Texturen begrenzt. Es wird allerdings davon ausgegangen, dass die meisten Details nicht relevant für eine ordnungsgemäße Bestimmung der Geokoordinaten sind, da die grundsätzlichen Formen (Torbögen, Fenster etc.) vorhanden sind, anhand derer eine Bestimmung erfolgen könnte. Weiter ist die Kathedrale im Allgemeinen in der Realität von allen Seiten gut erreichbar und kann auch aus weiterer Entfernung (außer die Nordfassade) gut fotografiert werden. Somit bieten sich trotzdem genügend Motive für ein synthetisches Training an.



(a) Südfassade real[20] vs. 3D-Modell



(b) Nordfassade real[21] vs. 3D-Modell

(c) Westfassade real[22] vs. 3D-Modell

(d) Ostfassade real[23] vs. 3D-Modell

Abbildung 3.6: Vergleich der realen Notre Dame mit dem 3D-Modell

[23]Von DXR / Daniel Vorndran, CC BY-SA 3.0. https://commons.wikimedia.org/w/index.php?curid=36208575 (Letzter Zugriff: 17.02.2021)

[23]Quelle: Google Street View. https://goo.gl/maps/DFz9YUrAkWmBkpQV6 (Letzter Zugriff: 28.03.2021)

[23]Von DXR / Daniel Vorndran, CC BY-SA 3.0. https://commons.wikimedia.org/w/index.php?curid=36208575 (Letzter Zugriff: 28.03.2021)

[23]Von Uoaei1 - Own work, CC BY-SA 4.0. https://commons.wikimedia.org/w/index.php?curid=45384120 (Letzter Zugriff: 28.03.2021)

Umgebung Um das Trainingsmodell möglichst robust gegenüber externer Einflussfaktoren der Umgebung zu machen, wurden die von Tremblay et al. [72] angesprochenen Variationen wie sich ändernde Lichtverhältnisse und zufällig wechselnde Hintergründe in den Erstellungsprozess integriert. Die Nutzung unterschiedlicher Lichteinfälle, Farben und Intensitäten sorgen für ein variables Erscheinungsbild der Textur des 3D-Modells, während die wechselnden Hintergründe das neuronale Netz dazu zwingen sollen, sich auf das Modell zu konzentrieren und den Hintergrund weitestgehend zu ignorieren. Mit großen Trigger Volumes wurden schließlich die Distraktoren unterschiedlicher Größe und Form zufällig eingeblendet (Abbildung 3.7).

Abbildung 3.7: Synthetischer Datensatz mit Distraktoren und zufälligem Hintergrund

3.2.5 Erkenntnisse aus Vorarbeiten

In [31] wurde ein erster synthetischer Datensatz mit Fotos der Auflösung von 256 x 256 Pixeln mit vergleichbarem geografischen Radius und Anzahl der Samples des realen Datensatzes erstellt, um unter ähnlichen Bedingungen einen Vergleich mit diesem durchführen zu können. Dafür wurde ein Synchronisations-Mechanismus implementiert, mit dem die Fotos beider Datensätze in die gleichen Klassen eingeteilt wurden. Tabelle 3.2 zeigt sowohl den realen Datensatz *Real18.600* nach Anwendung der Klassengrößenbegrenzung auf minimal 50 und maximal 600 Fotos (vgl. Tabelle 3.1) und nach der Synchronisation mit dem synthetischen Datensatz (*Real18.600.synced*) als auch einen synthetischen Datensatz ohne (*Synth18.600*) und mit Synchronisation (*Synth18.600.synced*). Durch die Synchronisation hat sich der synthetische Datensatz von 128 möglichen Klassen auf 52 reduziert, wodurch auch eine bedeutende Menge Trainingssamples verworfen wurden. Die Beschränkung hatte allerdings keine Auswirkungen auf die Trainingsgenauigkeit und sogar einen positiven Effekt auf die Validierung. Die Vermutung liegt hier nahe, dass die Anzahl der Klassen im Falle von *Real18.600* und besonders *Synth18.600* zu hoch für die entsprechende Zahl der Samples ist, da bei einem Multiclass-Klassifikator die Anforde-

rung an den Datensatz mit der Anzahl der Klassen steigt [18]. Des Weiteren wurde eine zweite Variante des synthetischen Datensatzes trainiert, jedoch ohne den Einsatz zufälliger Hintergrundbilder. Dadurch konnte die Validierungsgenauigkeit um fast 10 % gesteigert werden. Dies zeigt, dass die Komplexität des Datensatzes das Training erschwert. Da diese Komplexität jedoch nötig ist, damit die Realitätslücke geschlossen werden kann, wird dies als weiteres Argument gewertet, die Größe des Datensatzes zu erhöhen.

Datensatz-Variante	Samples (Train / Test)	Anz. Klassen	Max. Acc.	Max. Val. Acc.	Epochen
Real18.600	14251	65	0.9143	0.1373	30
Real18.600.synced	12368	52	0.9973	0.1704	50
Synth18.600	19469	128	0.993425	0.732334	50
Synth18.600.synced	11047	52	0.992483	0.775420	50

Tabelle 3.2: Trainings-Vergleich realer vs. synthetischer Datensatz

Aufgrund der Vielzahl an Parametern, die bei der Erstellung eines Datensatzes eine entscheidende Rolle spielen, erfolgte die Erzeugung weiterer Variationen im Kontext dieser Arbeit als iterativer Prozess im Sinne des Prozesses der *Knowledge Discovery in Databases (KDD)* [14]. Auf Basis von [22] und der Vermutung, dass die gewählte Auflösung für die Komplexität des dargestellten Objektes nicht ausreichen wird, wurden alle weiteren Datensätze mit einer Auflösung von 512 x 512 Pixeln erzeugt.

3.2.6 Verteilung

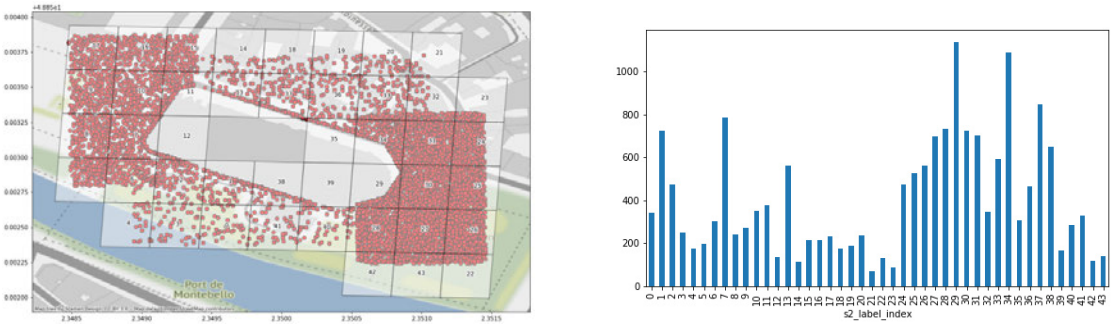
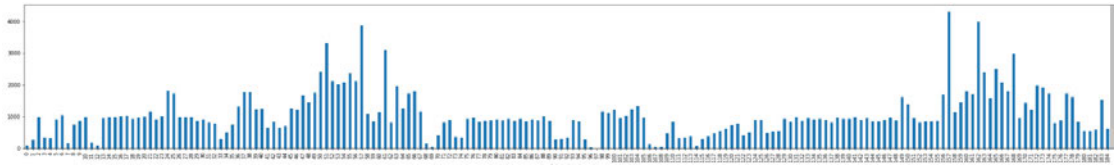


Abbildung 3.8: Verteilung synthetischer Bilder bei 17.735 Samples auf 44 Klassen

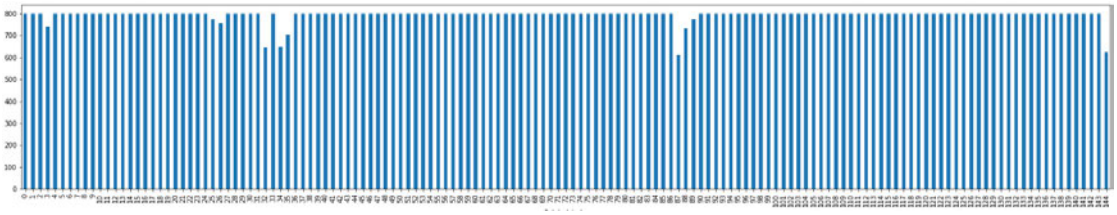
Durch Berechnung von Geopositionen jedes exportierten Bildes ist eine Darstellung der Verteilung der Fotos auf der Karte wie z. B. in Abbildung 3.1a auch für die synthetischen Daten möglich. In [31] wurde bereits auf eine ungleiche Verteilung der Daten auch bei Erstellung mit der Unreal Engine hingewiesen. Für weitere Experimente in dieser Arbeit

wurden deshalb einige weitere Datensätze und Datensatzvariationen erstellt, um die Auswirkungen variabler Datensatzgrößen, Verteilung der Samples und Anzahl der Klassen auf das Training zu erforschen.
Abbildung 3.8 zeigt die Verteilung von 17.735 generierten Samples auf 44 Klassen. Anhand dieses Beispiels lassen sich zwei Punkte gut erläutern: Dichte Ballungsgebiete lassen sich auf unterschiedliche Größen der Camera Volumes bei gleicher Gesamtzahl von erstellten Fotos pro Volume zurückführen. Mit Größe und Anzahl der Volumes könnte somit die Abdeckung bestimmter Perspektiven, aus denen in der Realität besonders viele Motive erstellt werden könnten, erhöht werden. Allerdings lässt sich auch eine Unregelmäßigkeit bei der statistischen Verteilung der generierten Fotos erkennen: Besonders an den äußeren Rändern des Gebäudes wurden offensichtlich verhältnismäßig viele Fotos angefertigt. Dieser Umstand sorgt u. a. dafür, dass die Klasse mit der höchsten Abdeckung (Klasse 29) 1.135 Samples enthält, obwohl sie nicht einmal vollständig abgedeckt ist. Selbst bei einem definierten Maximum von z. B. 800 Fotos pro Klasse wäre die Differenz zu den Minima sehr groß. Der Grund erscheint nicht naheliegend, da jedoch an dieser Stelle die Bewegung der Kamera blockiert wird. Um nicht in das Modell „hineinzufahren“, wurden dort möglicherweise die Fotos erstellt, die ursprünglich im Bereich des Gebäudes gelegen hätten. An dieser Stelle könnte eine genauere Untersuchung der Arbeitsweise des NDDS-Plugins erfolgen, die allerdings den Rahmen dieser Arbeit sprengen würde.

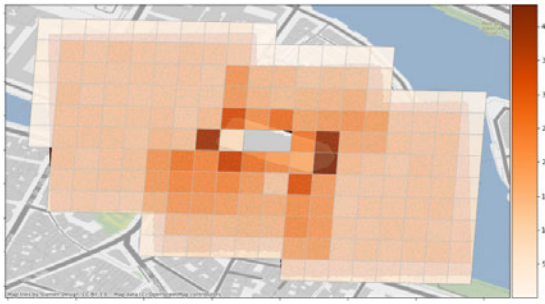
Stattdessen wurden bei der Erstellung der Camera Volumes im nächsten Schritt die Gesamtzahl der pro Volume aufzuzeichnenden Samples an den Flächeninhalt der Volumes angepasst. Gleichzeitig wurde die Anzahl der Gesamtsamples auf 200.000 erhöht, von denen letztendlich 195.495 fehlerfreie Dateien genutzt werden konnten. Abbildung 3.9 zeigt die Verteilung dieser Samples auf 185 Klassen ohne eine Begrenzung der Minimal- und Maximalwerte von Samples pro Klasse (Abbildung 3.9a und 3.9c) sowie nach Beschränkung auf minimal 600 und maximal 800 Samples pro Klasse (Abbildungen 3.9b und 3.9d). Wie zu sehen ist, kann dadurch eine optimale Verteilung bei Beschränkung der Klassenanzahl auf 145 erreicht werden. Dies geht allerdings auf Kosten vieler Samples, denn der Trainingsdatensatz besteht nach der Anpassung aus nur noch aus 115.015 Samples. Da die Generierung per Domain Randomization allerdings sehr effizient bei niedrigen Systemanforderungen durchgeführt werden kann, stellt dies kein Problem dar. Zudem lassen sich aus diesem Datensatz weitere kleine Datensätze mit individueller Klassenauswahl erstellen und damit verschiedene Trainings durchführen.



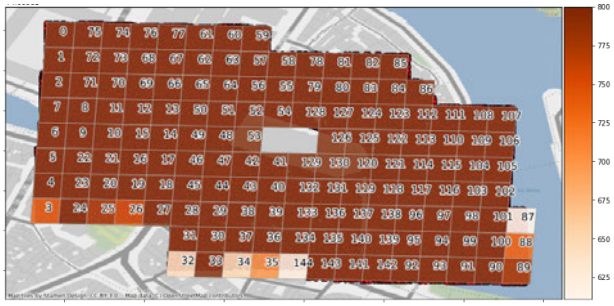
(a) Verteilung auf 185 Klassen vor Begrenzung



(b) Verteilung auf 145 Klassen nach Begrenzung auf min. 600 und max. 800 Samples pro Klasse



(c) Verteilung vor Begrenzung



(d) Verteilung nach Begrenzung

Abbildung 3.9: Verteilung synthetischer Bilder bei 195.495 Samples

4 Experimente

In diesem Kapitel werden einige Versuche durchgeführt, um die Verwendung realer Fotos und synthetisch erzeugter Datensätze zu kontrastieren. Grundlage der Experimente sind mit der Deep Learning Architektur InceptionV3 durchgeführte Trainings mit unterschiedlichen Kombinationen von Datensätzen, deren Erstellung in Kapitel 3 beschrieben wurde. In [30] und [31] wurde eine Pipeline zur Rekonstruktion der Kameraposition aus Internetfotos implementiert, die für diese Experimente als Grundlage dient und dafür erweitert wurde, weshalb sie hier zu Beginn kurz beschrieben wird. Aufgrund der Vielzahl von Möglichkeiten, die Bedingungen der Trainings zu ändern, wird zuerst ein Überblick über die im weiteren Verlauf folgenden Experimente durchgeführten Parametereinstellungen gegeben. Zu beachten ist, dass es sich bei diesen Experimenten vordergründig um eine prototypisches Herantasten innerhalb dieser Thematik handelt. In Kapitel 5 werden deshalb anschließend einige Verbesserungsmöglichkeiten aufgezeigt.

4.1 Vorbereitungen

4.1.1 Konzept der Pipeline

Die implementierte Pipeline lässt sich kurz in die vier Schritte Datenbeschaffung bzw. -erzeugung, Datenverarbeitung, Training und Analyse einteilen. In Kapitel 3 wurde das Konzept der Datenbeschaffung und Datenverarbeitung beschrieben. Die Implementierung erfolgte über eine Python Anwendung, die die mit der Unreal Engine erzeugten synthetischen Fotos einliest, in Geokoordinaten aus Längen- und Breitgrad umrechnet und mithilfe der S2 Geometry Library in ein Raster einteilt. Um die Verteilung der Fotos für die Analyse mit einer Vielzahl von Anwendungen visualisieren zu können, wurden Geoposition und Raster mit der GeoPandas[24] Bibliothek in das GeoJson Format exportiert. Für das Training wurde die Keras API in Verbindung mit Tensorflow genutzt.

[24]GeoPandas Bibliothek: https://geopandas.org/ (Letzter Zugriff: 22.02.2021)

Keras wiederum beinhaltet ein vollständiges Modell des Inceptionv3-Architektur, das sowohl vortrainiert mit dem ImageNet Datensatz als auch ohne Gewichte geladen werden kann. Für die folgenden Datensätze wurde die Architektur vollständig trainiert, da das Training mit vortrainierten Gewichten nicht den gewünschten Effekt erzielen konnte. Hier wären noch weitere Analysen zur Ursachen-Findung und dadurch ggf. verbesserter Ergebnisse nötig. Als Optimizer wurde der Stochastic Gradient Descent (SGD) mit einer Lernrate von 0.01 und als Loss-Funktion die Categorical Crossentropy genutzt, die sich besonders für Multiclass-Klassifikatoren eignet.

Die Trainingsbilder wurden in ihrer „rohen“ Form dem Modell zur Verfügung gestellt. Das bedeutet, dass ein Ordner im Dateisystem beliebig viele Paare aus Bild-Datei und Json-Datei für die Metadaten enthält. Aufgrund der hohen Datenmenge (siehe Abschnitt 3.2) wurde die Auswahl der nach der Einteilung der Samples auf die Klassen tatsächlich für das Training genutzten Fotos und der anschließenden Trennung in Trainings- und Validierungsmenge im Pandas `Dataframe`-Datentyp gespeichert. Diese enthalten die Dateipfade und werden für das Training über Keras' `DataGenerator`-Instanzen bei Bedarf aus dem Speicher geladen. So kann ein Datensatz flexibel für beliebige Datensatzvariationen mit unterschiedlicher Klassenaufteilung genutzt werden.

4.1.2 Eingrenzung des Experiment-Umfangs

Wahl des Trainingsmodells

Aus der Analyse (Abschnitt 2.6) ging hervor, dass die Wahl der Netzarchitektur des trainierenden Modells entscheidenden Einfluss auf die Ergebnisse haben kann. Um dies an einem kleinen Fallbeispiel zu zeigen, wurde die Minimalarchitektur aus [30] in der darauffolgenden Arbeit [31] durch die InceptionV3-Architektur ersetzt und jeweils mit drei unterschiedlichen Datensätzen trainiert.

Abbildung 4.1 zeigt die Trainingsverläufe dreier unterschiedlicher Datensätze, die sowohl mit der Minimalarchitektur als auch der InceptionV3-Architektur trainiert wurden. Auf die unterschiedlichen Datensätze und Gründe für die Differenz der Validierungsgenauigkeit wird im weiteren Verlauf dieses Abschnitts eingegangen. Aufgrund des deutlich besseren Abschneidens der InceptionV3-Architektur werden die folgenden Experimente nur noch mit dieser Architektur trainiert, um den Fokus auf die Eigenschaften des Datensatzes zu lenken.

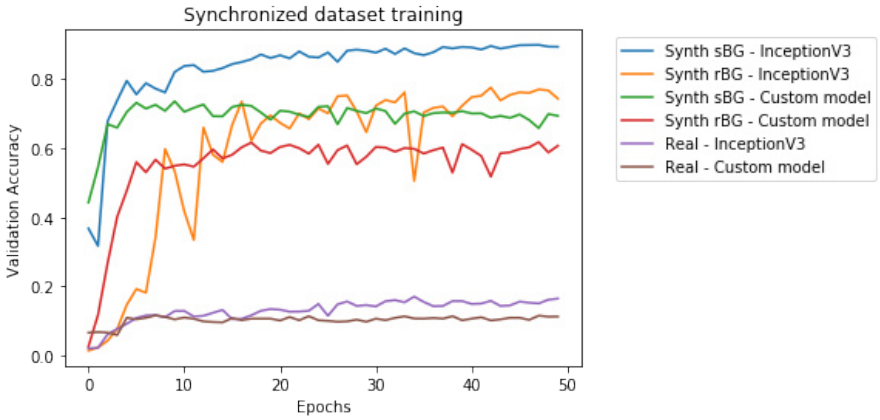


Abbildung 4.1: Vergleich eines Trainings mit minimalistischer DL-Architektur und InceptionV3-Architektur mit verschiedenen Datensätzen (aus [31])

Labelanpassungen der Datensätze

Zur Erinnerung: In Kapitel 3 (Datenmaterial) wurden sowohl Datensätze mit realen Fotos aus Flickr sowie einige synthetische Datensätze mit der Unreal Engine erzeugt. Als Metadaten wurden lediglich die Geopositionen der Fotos als Tupel aus Längen- und Breitengrad gespeichert. Die Labels, die für das Training benötigt werden, also die Einteilung der Fotos in die S2-Zellen, erfolgt allerdings dynamisch vor Beginn eines jeden Trainings anhand voreingestellter Parameter.

Diese umfassen die

- minimale Anzahl an Fotos pro Zelle, die
- maximale Anzahl an Fotos pro Zelle, die
- Größe der S2-Zellen und die
- Gesamtzahl der Zellen und damit der Klassen.

Ein Datensatz kann also für mehrere Experimente bei gleichbleibenden Hyperparametern des neuronalen Netzes verwendet werden, ohne den Inhalt der Bilder zu ändern. Stattdessen erfolgt nur eine Anpassung der Labels und dadurch bedingt auch die Gesamtzahl der Samples.

Erhöhung der Sample-Auflösung

Eine Erhöhung der Auflösung der Fotos von ursprünglich 256 x 256 Pixel, die für den realen Datensatz verwendet wurde, auf 512 x 512 Pixel wurde lediglich beim ersten synthetisch erstellten Datensatz durchgeführt. Alle folgenden Kombinationen wurden durchweg mit 512 x 512 Pixeln durchgeführt. Auf ein weiteres Training des realen Datensatzes mit einer besseren Auflösung wurde aufgrund der wenig erfolgversprechenden Ergebnisse in [31] und der daraus resultierenden Entscheidung, der Herausforderung mit synthetischen Daten zu begegnen, verzichtet.

Anpassung von Hyperparametern

Die in Kapitel 2 angesprochene Feinabstimmung von diversen Hyperparametern eines neuronalen Netzes haben in dieser Arbeit eine eher untergeordnete Rolle[25]. Da laut [18] und [72] die Qualität des Datensatzes in den meisten Fällen einen deutlich größeren Effekt auf das Training hat (vgl. Kapitel 3), soll durch die folgenden Experimente primär dieser Einfluss beobachtet werden.

Pretraining

In Abschnitt 2.2.2 wurde das Transfer Learning als Möglichkeit vorgestellt, mit dem die Gewichte von Modellen übernommen werden, die ursprünglich für eine andere Domäne trainiert wurden (z. B. auf dem ImageNet Datensatz). Eine Möglichkeit ist, die Gewichte aus den vorderen Schichten, die für die Merkmal-Extraktion zuständig sind, während des Trainings „einzufrieren“ und somit nur den Klassifikator-Teil des Netzes mit den neuen Daten zu trainieren [25]. Getestete Trainings mit den in dieser Arbeit synthetisch erzeugten Daten haben allerdings deutlich schlechtere Ergebnisse geliefert, sodass darauf im Weiteren verzichtet und stattdessen das gesamte Modell trainiert wurde. Dieses Verhalten konnten Tremblay et al. [72] ebenso beobachten und vermuten, dass die Vielfalt des domänenrandomisierten Datensatzes ein vollständiges Training benötigt, um die Gewichte der Situation vorteilhaft anpassen zu können.

[25] Im Verlaufe einiger Trainings wurden zwar verschiedene *Lernrate* und *Batch Size* Anpassungen vorgenommen, diese hatten jedoch nur geringen Einfluss auf die Ergebnisse, sodass die besten Einstellungen für alle weiteren Trainings beibehalten wurden.

Realer Datensatz

Bei den im bisherigen Verlauf erwähnten Experimenten, die bereits mit dem realen Datensatz durchgeführt wurden (siehe Tabellen 4.1 und 4.1), handelt es sich ausschließlich um statische Einteilungen mit dem S2-Level 18. Um auszuschließen, dass entgegen der Erwartungen eine Kombination gefunden werden kann, die ein annehmbares Trainingsergebnis erzielt, wurden testweise Trainings mit der dynamischen (siehe Abbildung 3.3a) und statischen Zelleinteilung mit dem S2 Level 17 durchgeführt. Tabelle 4.1 zeigt zum einen, dass durch die verschiedenen Einteilungen eine hohe Diversität der Trainingsbedingungen entsteht. Obwohl bei allen Trainings eine Genauigkeit von über 90 % erreicht werden kann, liegt der Höchstwert der Validierungsgenauigkeit bei nur 18,01 %. Eine Interpretation dieser Ergebnisse ist aufgrund der Bedingungen kaum möglich und wäre rein spekulativ. Aufgrund der großen Differenz der Genauigkeiten zwischen Training und Validierung kann jedoch von einem sehr starken Overfitting ausgegangen werden. Bei Beachtung des in Kapitel 3 beschriebenen zum Teil willkürlichen Inhalts der Fotos ist dies nicht verwunderlich, sodass auf weitere Experimente mit diesem Datensatz verzichtet wird.

Datensatz-Variante	Gesamt-Samples	Anz. Klassen	Max. Acc.	Max. Val.-Acc.	Epochen
Real13.600dyn	29720	113	0.9969	0.1025	30
Real17	11843	44	0.9725	0.1801	30
Real18.600	14251	65	0.9143	0.1373	30
Synchronized	12368	52	0.9973	0.1704	50

Tabelle 4.1: Ergebnisse einiger Trainings mit realem Datensatz

Umkreis

Wie in Abschnitt 3.2 bereits angedeutet, ist der geografische Umkreis der synthetischen Datensätze für die bessere Vergleichbarkeit dem der realen Daten angelehnt. Zwar könnte problemlos ein größerer Umkreis synthetischer Fotos erzeugt werden, dies würde aber mehr Trainingsaufwand ohne großen Mehrwert bedeuten. Schließlich hilft es keiner Anwendung, geografische Bereiche zu simulieren, die in der Realität gar nicht begehbar sind und deshalb dort keine realen Fotos existieren. Unabhängig davon soll zuerst die generelle Machbarkeit erforscht werden, die auch mit kleineren Datensätzen möglich sein sollte. Idealerweise kann dann analysiert werden, bei welcher Entfernung zum Objekt das Modell keine sinnvollen Vorhersagen mehr treffen kann.

Wechselnde Hintergründe

Im in [31] durchgeführten Vorab-Experiment, auf das bereits zum Teil in Abbildung 4.1 Bezug genommen wurde, wurde der synthetische mit den Klassen des realen Datensatzes synchronisierte Datensatz sowohl mit statischen (*Synth sBG)*, blaue Kurve) als auch dynamischen Hintergrund (*Synth rBG*, orange Kurve) trainiert. Hier ist zu erkennen, dass die Validierungsgenauigkeit beim Datensatz mit statischem Hintergrund mit einem Maximalwert von 89 % deutlich genauer und stetiger ausfällt als beim zufälligen Austauschen der Hintergründe (Maximalwert 76 %). Aufgrund des positiven Effekts des Schließens der Realitätslücke (vgl. Abschnitt 2.5.2), wurden nachfolgende Experimente allerdings durchweg mit randomisierten Hintergründen durchgeführt, um bessere Ergebnisse beim Testing mit realen Daten anzustreben.

4.1.3 Erstellung eines Testdatensatzes

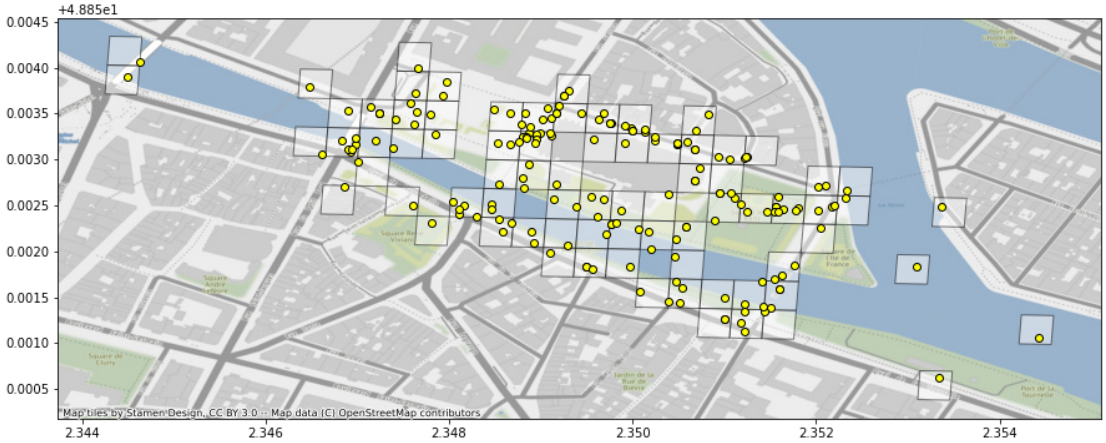


Abbildung 4.2: Testdatensatz mit 182 Testfotos

Um die Qualität des trainierten Modells aussagekräftig bewerten zu können, reicht es generell nicht aus, sich auf die Validierungsergebnisse zu stützen (siehe Abschnitt 2.4.1). Besonders in diesem Fall wurde mit dem Training rein synthetischer Daten eine Domäne geschaffen, die mit der Realität nur bedingt zu vergleichen ist. Wie die Beschreibung des Bildinhalts der synthetischen Daten schon gezeigt hatte, ist der einzige Vergleichswert zwischen der synthetischen und realen Domäne die Darstellung des 3D-Modells der Notre Dame, wenngleich sich dieses in einigen Punkten leicht und zum Teil schwer unterscheidet (Abschnitt 3.2.4). Die trainierten Modelle müssen somit gegen reale Daten

getestet werden, die das Modell während des Trainings noch nicht gesehen hat. Zudem muss sichergestellt sein, dass korrekte Ground Truth Label vorhanden sind. Ein realer Datensatz von der Qualität wie unter 3.1 beschrieben kann somit nicht verwendet werden – schließlich stellt dieser das Eingangsproblem dieser Arbeit, fehlerhafte Metadaten korrigieren zu wollen, dar.

Um trotzdem eine grobe Einschätzung über die Funktionalität der trainierten Modelle in der Realität treffen zu können, wurde einige Screenshots aus Google Street View erstellt. Aufgrund der 360°-Ansichten bietet Street View die Möglichkeit, bei gleichbleibender Geoposition Anpassungen am Blickwinkel der Kamera vorzunehmen. Die Umgebung der Notre Dame bietet in diesem Fall zusätzlich den besonderen Vorteil, dass sie von allen Seiten begehbar und befahrbar ist. Deshalb bietet der Street View Datensatz eine sehr gute Abdeckung. Dieser Vorgang wurde manuell durchgeführt und soll vorerst nur zum prototypischen Testen dienen. Entsprechende Programmierschnittstellen bieten jedoch auch automatisierte Möglichkeiten, größere Datenmengen zu extrahieren. Hier wurden stattdessen zur Anreicherung einige Fotos aus dem realen Datensatz aus 3.1 übernommen, bei denen bei manueller Sichtung von korrekten Labels ausgegangen werden kann und der Testdatensatz durch eigene Urlaubsfotos mit Geopositionen ergänzt. Es entstand ein Datensatz mit 182 Testfotos, mit denen maximal 77 Klassen getestet werden können (Abbildung 4.2). Je nach trainiertem Radius des zu testenden Modells kann die Größe des Datensatzes abweichen. Die entsprechenden Klassen für die sechs Testfotos in den Randbereichen wurden von keinem der folgenden Modelle trainiert, sodass der maximal abgedeckte Bereich etwa 100.000 m^2 entspricht. Generell wurde versucht, möglichst viele von Personen begehbare Orte in den Datensatz zu integrieren, sodass keine großen Anhäufungen gleichartiger Fotos existieren. Eine Ausnahme bildet der Bereich des Haupteingangs an der Westfassade der Kathedrale: Hier sollen neben der Totalaufnahme des Gebäudes zusätzliche Detailaufnahmen der komplexen Eingangstore mit den zahlreichen Figuren getestet werden.

4.2 Erster Durchgang

4.2.1 Datensatz-Erzeugung

Aufgrund der vielen Variablen bei der Datenerzeugung und der Ungewissheit, wie viele Samples tatsächlich benötigt werden, um akzeptable Ergebnisse zu erzielen, erfolgte für

die weitere Entwicklung eines geeigneten Datensatzes ein iterativer Ansatz aus Datenerstellung und Training, um sich experimentell der Fragestellung zu nähern, welche Faktoren das Training beeinflussen. Neben der (einmaligen) Erhöhung der Sample-Auflösung wurden Datensatz-Variationen unterschiedlicher Anzahl der Samples, Klassen, Klassengröße und Verteilung der Fotos angefertigt. Die Trainingsdauer wurde generell für maximal 50 Epochen angesetzt, in einigen Fällen aber auch bereits früher abgebrochen, wenn keine entscheidenden Verbesserungen der Genauigkeiten mehr zu erwarten waren. Hier werden die Trainings in zwei Durchgängen zusammengefasst, wobei im ersten experimentelle grundlegende Versuche und im zweiten die Erkenntnisse aus Analysen des ersten Durchgangs angewendet wurden. Die Versuche werden im Folgenden als „Modelle“ bezeichnet, um die Erstellung der Datensätze und die anschließend damit trainierten Modelle zusammenfassend beschreiben zu können. Tabelle 4.2 zeigt die Modelle, die hier besprochen werden. Zum Vergleich mit den Ergebnissen aus dem vorherigen Projekt [31] sind ebenso der dort erstellte und anschließend mit dem realen Datensatz synchronisierte Datensatz in einer 256-Pixel-Ausführung (*Synth18.600* und *Synth18.600.synced*) aufgelistet. Das Modell *M-S1-512* deckt in etwa den geografischen Bereich von *Synth18.600* ab und ist dessen 512-Pixel-Ausführung, die ebenso zusätzlich mit den Klassen des ursprünglichen realen Datensatzes synchronisiert wurden (Modell *M-S1-512s*). Die Modelle sind mit den Bezeichnungen *S1* bis *S3* versehen und bestimmen grob die Größe des mit synthetischen Daten abgedeckten Umkreises in absteigender Reihenfolge. Bei der Anzahl der Klassen werden zudem die Minimal- und Maximalbeschränkungen von Fotos pro Klasse angegeben. Mit *M-S2-400* und *M-S2-800* wird die Auswirkung einer Änderung auf die Maximalbegrenzung bei gleicher Datensatz-Variation untersucht, während *M-S3* keiner Begrenzung nach oben unterliegt. Abbildungen zu den Verteilungen der Fotos auf die Klassen aller Modelle sind im Anhang (Abbildung A.1) zu finden.

4.2.2 Training

Abbildung 4.3 zeigt die Trainingsverläufe der soeben vorgestellten Modelle und Tabelle 4.2 die Maximalwerte der jeweiligen Trainingsgenauigkeit („Max. Acc.“) und Validierungsgenauigkeit („Max. Val.-Acc.“). Grundsätzlich ist hier zu erkennen, dass die Trainingsgenauigkeit in allen Fällen (ausgenommen *M-S1-512*[26]) nach spätestens 20 Epochen bereits bei über 90% liegt. Bei steigender Anzahl der Klassen steigt die Kurve erwartungsgemäß langsamer. Eine Auswirkung der Sample-Anzahl auf den Trainingsverlauf kann

[26] Eine höhere Genauigkeit wäre bei einem längeren Training zu erwarten gewesen

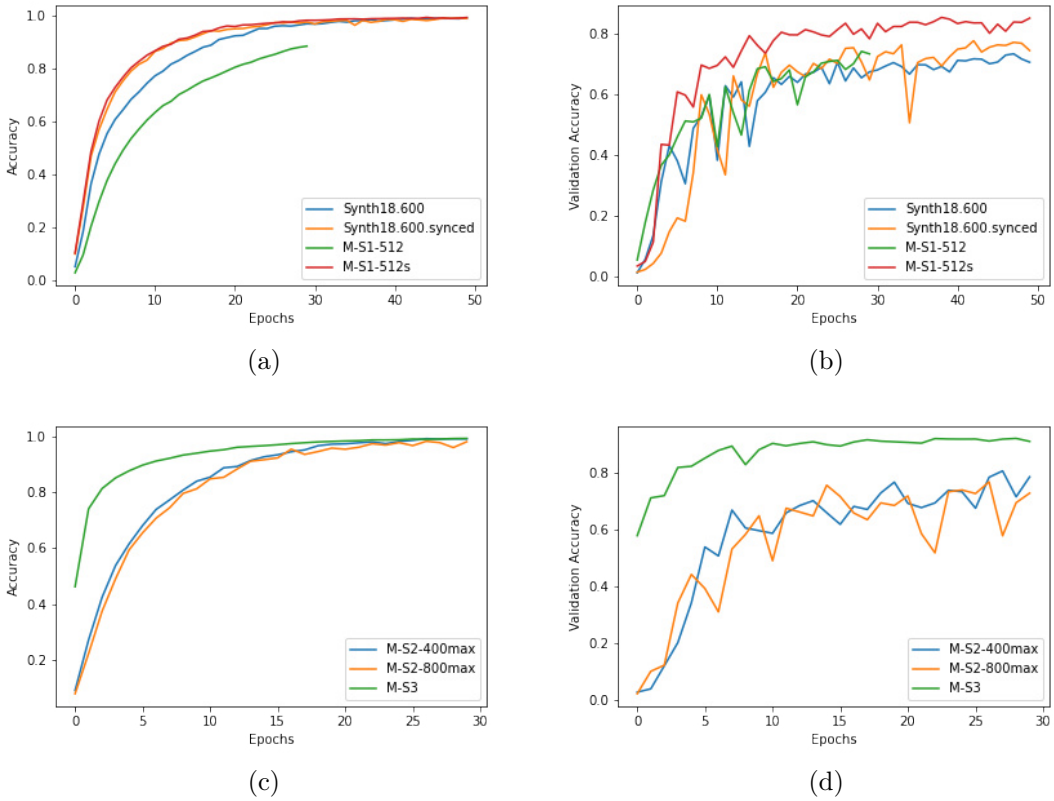


Abbildung 4.3: Trainingsverläufe des ersten Durchgangs

hier nicht erkannt werden, was besonders am Datensatz *M-S3* deutlich wird. In diesem Fall geht jedoch eine starke Ungleichheit der Verteilung der Samples auf die Klassen aus (siehe Differenz zwischen Klasse 5 und Klasse 7 in Abbildung A.1d). Die Genauigkeit ist dadurch als Metrik nicht primär geeignet, da sie die Leistung auf die Gesamtheit aller Klassen berechnet und Ungleichheiten vernachlässigt [22].

Der Verlauf der Validierungsgenauigkeit fällt hingegen sehr unterschiedlich aus, was auf die deutlichen Unterschiede des Informationsgehalts der Datensätze zurückzuführen sein könnte. Es gestaltet sich schwierig, aus diesen Ergebnissen logische Schlussfolgerungen abzuleiten. Was jedoch deutlich erkennbar ist, sind die verhältnismäßig stabilen Kurven und guten Maxima der Datensätze *M-S1-512s* und *M-S3*. Im ersten Fall könnte dies im direkten Vergleich mit *Synth18.600.synced* auf die Erhöhung der Pixel und Anzahl der Samples hindeuten. Im zweiten Fall werden viele Fotos auf wenig Klassen trainiert, was vermutlich dem Idealfall entspricht. Zudem kann man erkennen, dass besonders die

Datensätze *M-S2-800max* und *M-S2-400* selbst im Bereich der 30. Epoche eine hohe Fluktuation zeigen. Dies lässt intuitiv auf viele Zufallsschätzungen des Netzes schließen, was an der geringen Anzahl Samples liegen könnte. In die Entwicklung von *M-S1-512* könnte interpretiert werden, dass bei längerem Training die Fluktuation der Validierungsgenauigkeit nachlassen und allgemein weiter ansteigen wird. Aufgrund der hohen Anzahl der Klassen werden wahrscheinlich deutlich mehr Epochen benötigt, was an dem flachen Anstieg der Trainingsgenauigkeit abzulesen ist.

Modell-Bezeichnung	**Eigenschaften**	**Gesamt-Samples**	**Anz. Klassen (min. Fotos / max. Fotos)**	**Max. Acc.**	**Max. Val.-Acc.**	**Epochen**
Synth18.600	256px	19469	128 (50/400)	0,9934	0,7323	50
Synth18.600.synced	256px, synchronisiert	9047	52 (50/600)	0,9924	0,7754	50
M-S1-512	512px	51529	195 (50/400)	0,8862	0,7404	30
M-S1-512s	512px, synchronisiert	20724	55 (50/600)	0,9950	0,8521	50
M-S2-400	nur Nahaufnahmen	5582	44 (50/800)	0,9909	0,8043	30
M-S2-800	nur Nahaufnahmen	5743	44 (50/400)	0,9816	0,7658	30
M-S3	nah, starke Anhäufung	84821	40 (50/max)	0,9875	0,9057	30

Tabelle 4.2: Ergebnisse einiger Trainings mit synthetischen Datensätzen und -variationen

4.2.3 Testing

In diesem Abschnitt werden die trainierten Modelle mit dem in Abschnitt 4.1.3 erstellten Testdatensatz auf ihre Anwendbarkeit in der Realität überprüft und analysiert. Die Testdaten werden den trainierten Klassen des jeweiligen Modells zugeordnet und erhalten deshalb auch die Modell-spezifischen Indizes, die während des Trainings als Labels verwendet wurden. Befindet sich ein Testfoto in einem Bereich, der vom Modell nicht abgedeckt wurde, wird es nicht verwendet. Somit unterscheidet sich die Darstellung des Testdatensatzes je nach Modell leicht und wird deshalb im Folgenden pro Modell dem Testergebnis grafisch gegenübergestellt. Als Bewertungskriterien dient die Anzahl der exakt vorhergesagten Zellen. Diese kann allerdings kaum weniger Metrik zur Bewertung der Modelle verwendet werden, da durch die Verwendung von Zellen stellvertretend für Geopositionen eine diskrete Zielmenge definiert wurde. Für eine bessere Genauigkeit sollten die Geopositionen wie in der Realität als kontinuierliche Werte betrachtet werden können.

Zusätzlich wird folglich die Geoposition der Mitte der vorhergesagten Zelle bestimmt und der Abstand zur Mitte der Ursprungszelle gemessen. Beträgt diese Distanz weniger als 50 Meter, wird dies als Erfolg gewertet. Mit dem Wert 50 m können alle direkten Nachbarzellen in die Wertung einbezogen werden (Abbildung 4.4). Damit wird das Problem

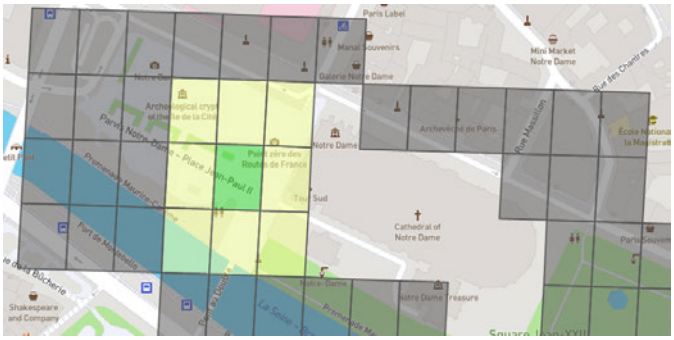


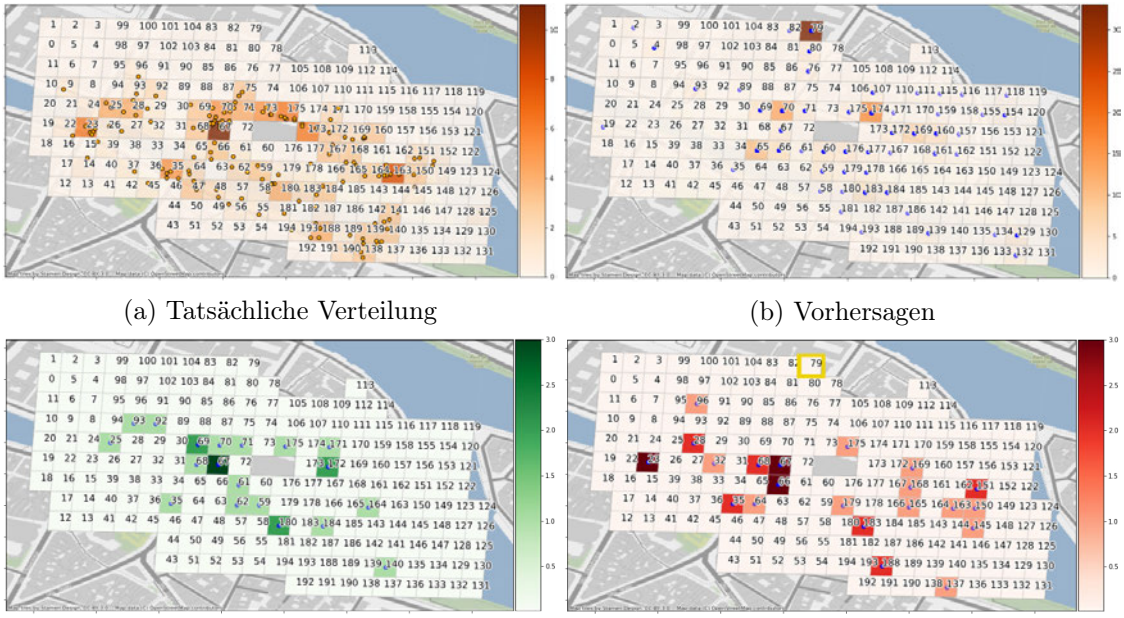
Abbildung 4.4: 50 Meter-Radius als Toleranzregion (gelbe Markierung)

gelöst, dass nahezu identische Bilder, die in den Grenzregionen zwischen zwei Zellen liegen, trotzdem in beiden Fällen mit in die Positivwertung einfließen können. Mit dieser Problematik wird sich im Kapitel 5 („Evaluation“) näher auseinandergesetzt. Für Details sei deshalb darauf vorab verwiesen.

Modell M-S1-512

Zum Testen des Modells *M-S1-512* stehen 173 Testfotos verteilt auf 70 Klassen zur Verfügung, von denen drei exakt richtig bestimmt werden und somit nur 1,73 % des Datensatzes ausmachen. Bei Einbeziehung der 50 m-Distanz können mit diesem Modell 24 Fotos erfolgreich bestimmt werden, was einem Erfolg in Relation zur Testdatenmenge von 13,87 % bei einer durchschnittlichen Vorhersage-Wahrscheinlichkeit von 44,34 %, also der Wahrscheinlichkeit für einen Erfolgstreffer, entspricht.

Abbildung 4.5 stellt die tatsächlichen Positionen der Testfotos im Raster des Modells *M-S1-512* (links) im direkten Vergleich mit den Vorhersagen (rechts) dar. Auf den ersten Blick ist hier zu erkennen, dass die Anhäufung der Testfotos am Haupteingang sowie dem südlichen Bereich vom Modell nicht erkannt werden, sondern nur einzelne Fotos in diesen Regionen vermutet werden. Die stärksten Anhäufungen finden sich in Zelle 79 mit 33 Fotos und Zelle 174 mit 16 Fotos. Diese Klassen wurden allerdings gar nicht vom Testdatensatz abgedeckt, sodass diese Vorhersagen nicht zutreffen. Eine Untersuchung der Ursprünge der Zelle 79 zugeordneten Fotos lässt keine Rückschlüsse zu, da sich die Fotos in allen Bereichen befinden, wie in Abbildung 4.5d zu sehen. Insgesamt wurden aber 19 von 70 möglichen Klassen zumindest einmal getroffen. Doch auch hier täuscht



(a) Tatsächliche Verteilung

(b) Vorhersagen

(c) Nur korrekte Vorhersagen

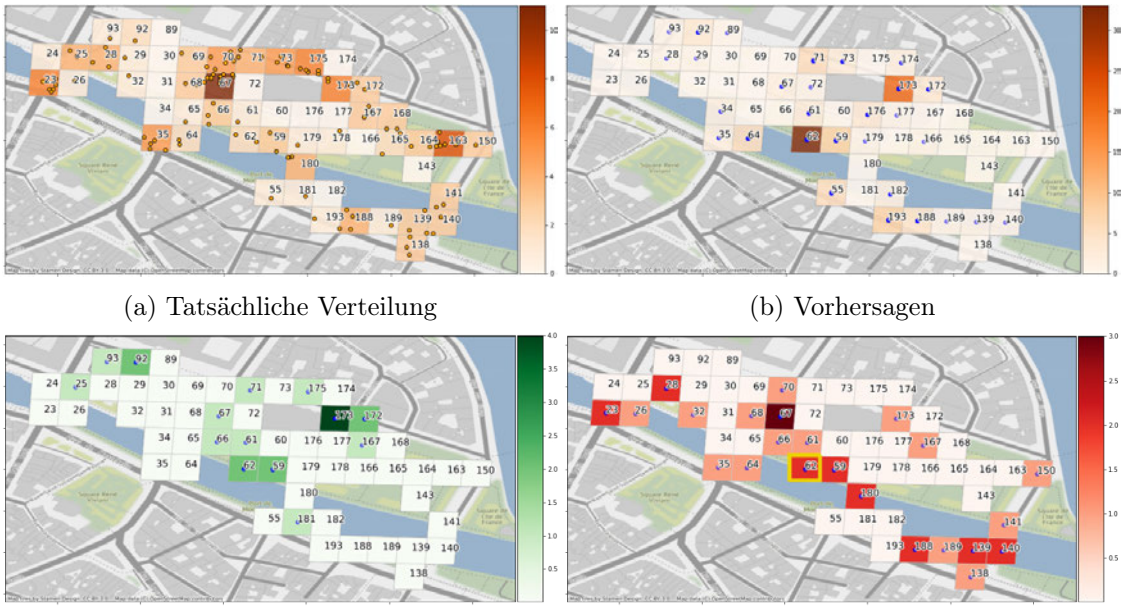
(d) Ursprünge der vorhergesagten Klasse 79

Abbildung 4.5: Vorhersagen der Testfotos mit Modell *M-S1-512*

die Darstellung: Gemessen an der Skala von Abbildung 4.5b werden 13 Fotos der Klasse 70 zugeordnet. Von denen können allerdings nur zwei als Erfolg gewertet werden, die ihren Ursprung in Klasse 69 hatten und somit ca. 25 m vom wahren Wert entfernt liegen. Abbildung 4.5c zeigt deshalb die tatsächlich im Toleranzbereich korrekt bestimmten Klassen aus der Sicht der echten Klassen ohne Falsch-Positive. Das bedeutet, dass die grün markierten Zellen entweder exakt oder durch eine der umliegenden Klassen bestimmt wurden.

Modell M-S1-512s

Die mit den Klassen des realen Datensatzes synchronisierte Variante des *M-S1*-Modells zeigt deutlich bessere Ergebnisse. In diesem Fall standen 132 Fotos verteilt auf 43 Klassen zum Test bereit, von denen 10 exakt richtig bestimmt wurden und der Test mit 7,58 % deutlich über dem Wert des *M-S1-512* Modells liegt. Bei der Berücksichtigung aller Vorhersagen unter 50 m Distanz zum echten Wert kommt das Modell auf eine Trefferquote von 21 Fotos (15,91 %) mit einer durchschnittlichen Vorhersage-Wahrscheinlichkeit von 68,90 %. Der Vergleich in Abbildung 4.6 zeigt, dass sich die Vorhersagen auch hier wieder auf zwei wesentliche Anhäufungen (Klassen 62 und 173) konzentrieren. Im Gegensatz zu



(a) Tatsächliche Verteilung

(b) Vorhersagen

(c) Nur korrekte Vorhersagen

(d) Ursprünge der vorhergesagten Klasse 62

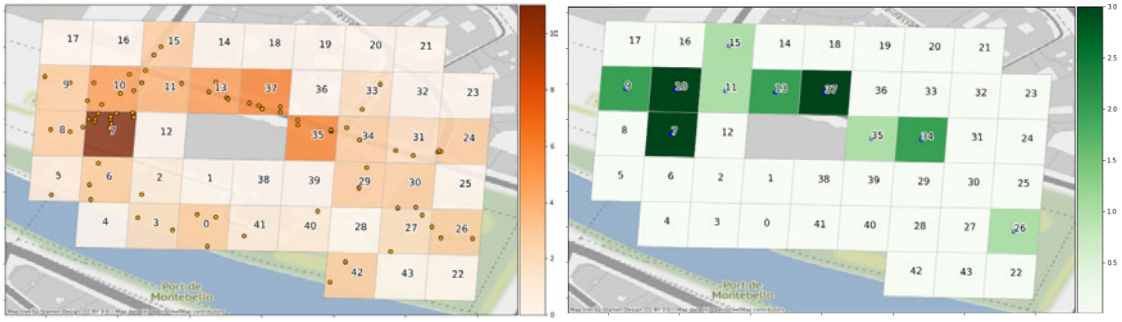
Abbildung 4.6: Vorhersagen der Testfotos mit Modell *M-S1-512s*

M-S1-512 sind diese jedoch nur in relevanten Bereichen zu finden, sodass dadurch auch mehr Treffer zu verzeichnen sind (Abbildung 4.6c). Eine Analyse der realen Fotos, die für Klasse 62 gehalten wurden, zeigt aber auch hier eine weite Streuung, wenngleich diese sich zumindest korrekterweise auf der Südseite des Objekts befinden (siehe Abbildung 4.6d).

Da die Modelle *M-S1-512* und *M-S2s-512* jeweils auf die vorherigen Modelle mit einer Auflösung von 256 x 256 Pixeln aus dem vorangegangen Projekt [31] basieren, wurde der Testdatensatz gegen die beiden Modelle *Synth18.600* und *Synth18.600.synced* getestet. Das Modell *Synth18.600* konnte nur 3,66 % der Fotos akzeptabel klassifizieren, was genau einem Viertel der Menge des Modells *M-S1-512* entspricht. Auf Klassenebene wurden insgesamt nur zwei Klassen korrekt identifiziert, somit rund einem Zehntel des höher auflösenden Modells. Bei den Modellen *Synth18.600* und *M-S1-512s* fällt der Unterschied deutlich geringer aus. Hier stehen der Vorhersage der Fotos 11,81 % und 15,91 % sowie 19,51 % und 32,56 % bei der mindestens einmaligen Erkennung von Klassen gegenüber. Diese Ergebnisse bestätigen allerdings die Vermutung, dass bessere Ergebnisse mit der Erhöhung der Sample-Auflösung und Erhöhung der Anzahl der Samples erreicht werden können. Weitere Experimente erfolgen deshalb ausschließlich mit einer Sample-Auflösung von 512 x 512 Pixeln.

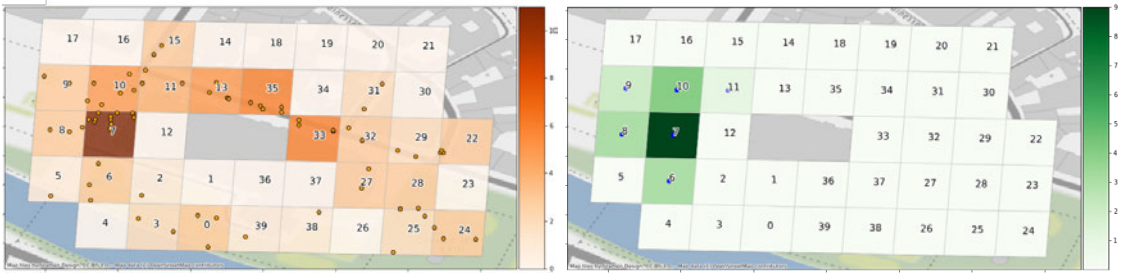
Modelle M-S2-400 und M-S3

Aufgrund der hohen Streuung der bisherigen Ergebnisse und der Tatsache, dass bei einer kleineren Klassenanzahl bessere Ergebnisse zu erwarten sind, wurden mit den Modellen *M-S2-400* und *M-S3* zwei Varianten getestet, die mit 44 bzw. 40 Klassen trainiert wurden. Zudem wurde *M-S2* mit insgesamt 5.743 Samples sehr klein und *M-S3* mit 84.821 Samples um ein Vielfaches größer angesetzt. Mit den Tests soll überprüft werden, inwiefern die Eigenschaften der unterschiedlichen Seiten des Notre Dame Modells erkannt werden und wie viele Fotos dafür nötig sind. *M-S3* wurde zudem mit einer Überrepräsentation von Samples in Klasse 7 trainiert.



(a) Tatsächliche Verteilung *M-S2-400*

(b) Nur korrekte Vorhersagen *M-S2-400*



(c) Tatsächliche Verteilung *M-S3*

(d) Nur korrekte Vorhersagen *M-S3*

Abbildung 4.7: Vorhersagen der Testfotos mit den Modellen *M-S2400* und *M-S3*

Abbildung 4.7 zeigt die Verteilung der Testlabels auf die übrigen Klassen bei diesen Modellen (Abb. 4.7a und Abb. 4.7c) und die richtig klassifizierten Testfotos in den Abbildungen 4.7b und 4.7d. Hier ist zu erkennen, dass das Modell *M-S2-400* insgesamt zehn Klassen und *M-S3* nur sechs Klassen korrekt bestimmt. Absolut gesehen werden allerdings 22 Fotos durch *M-S3* statt 19 durch *M-S2-400* korrekt bestimmt, von denen fünf statt einem Foto exakt bestimmt werden. Auffällig ist hierbei jedoch, dass die von *M-S3* vorhergesagten Klassen ausschließlich im Bereich des Haupteingangs der Notre Dame

lokalisiert sind und demnach dem überproportional durch Trainingsdaten abgedeckten Bereich entsprechen. Die Vermutung liegt hier nahe, dass sich das Modell diesem Bereich überangepasst hat und die anderen Klassen gar nicht mehr zur Geltung kommen. Auch wenn der prozentuale Anteil der erkannten Fotos im Falle von *M-S3* mit 27,85 % den bisher besten Wert darstellt, kann es aus den soeben beschriebenen Gründen nicht in der Praxis verwendet werden. Weiterhin ist die Entfernung der vorhergesagten Fotos unter 50 m vom realen Wert als Metrik im Vergleich zu den bisherigen Modellen ungenau, da hier aufgrund der deutlich niedrigeren Klassenanzahl die Wahrscheinlichkeit, falsche Bestimmungen zu treffen, geringer ist und das Modell dadurch statistisch aufgewertet wird.

Überblick

Mit den soeben vorgestellten Modellen, die in Tabelle 4.3 zusammengefasst sind, wurden experimentelle Versuche nach zwei Prinzipien durchgeführt: Zum einen wurden durch die Erhöhung von Auflösung und Anzahl der Samples die Modelle aus der vorhergehenden Arbeit aufgewertet und verglichen. Diese basierten darauf, möglichst den geografischen Bereich abzudecken, für den auch reale Fotos vorhanden sind. Aufgrund nicht zufriedenstellender Ergebnisse wurde zum anderen das geografische Einzugsgebiet weit eingeschränkt und der Versuch unternommen, dadurch zumindest die Himmelsrichtungen des Gebäudes zu erkennen. Abbildung 4.7 zeigt, dass dies mit beiden Modell nicht möglich ist, weshalb an dieser Stelle weitere Versuche nötig sind.

		Bestimmung der Testfotos			Bestimmung der Testklassen		
Modell-Bezeichnung	**ø Vorhersage-Wahrschein-lichkeit**	**Anzahl**	**Exakte Treffer: relativ / absolut**	**Distanz < 50 m: relativ / absolut**	**Anzahl**	**Exakte Treffer: relativ / absolut**	**Distanz < 50 m: relativ / absolut**
Synth18.600	47,95 %	164	1,22 % / 2	3,66 % / 6	64	3,12 % / 2	3,12 % / 2
Synth18.600.synced	66,26 %	127	4,72 % / 6	11,81 % / 15	41	9,76 % / 4	19,51 % / 8
M-S1-512	44,36 %	173	1,73 % / 3	13,87 % / 24	70	4,29 % / 3	27,14 % / 19
M-S1-512s	65,20 %	132	7,58 % / 10	15,91 % / 21	43	11,63 % / 5	32,56 % / 14
M-S2-400	56,00 %	82	1,22 % / 1	23,17 % / 19	25	4,00 % / 1	40,00 % / 10
M-S3	56,80 %	79	6,33 % / 5	27,85 % / 22	24	8,33 % / 2	25,00 % / 6

Tabelle 4.3: Testergebnisse des ersten Durchgangs

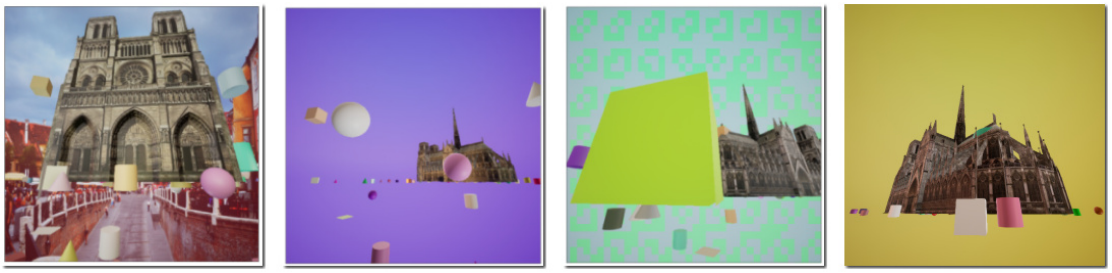
4.3 Zweiter Durchgang

4.3.1 Datensatzerzeugung

Die Testergebnisse im ersten Durchgang haben gezeigt, dass trotz einer guten Validierungsgenauigkeit des Trainings im Bereich von 90 % (bspw. im Fall des Modells *M-S3*) die Anwendung auf reale Daten keine vielversprechenden Ergebnisse liefert. Zudem gestaltet es sich schwierig, auf der Grundlage des kleinen Testdatensatzes und der hohen Varianz in den Eigenschaften der Modelle allgemeingültige Aussagen zu extrahieren. Ausreißern bei der Vorhersage wie im Modell *M-S1-512* (Abbildung 4.5b) oder lokale Fokussierungen durch Überanpassung wie in *M-S3* (Abbildung 4.7d) soll somit durch Masse in Form eines perfekt ausbalancierten Datensatzes mit hoher Anzahl Samples pro Klasse begegnet werden. Dafür wurde im nächsten Schritt das Modell *M-S4* erzeugt, das – bis auf wenige Ausnahmen an den Rändern – mit 800 Samples pro Klasse bei insgesamt 145 Klassen und 115.015 Samples den größten Datensatz darstellt (siehe Anhang Abbildung A.1e). An dieser Stelle sei bereits vorweggenommen, dass das anschließende Training (analog zu dem im ersten Durchgang angewendeten Prinzip) ebenso nicht die gewünschte Reaktion lieferte. Aufgrund dessen wurde eine analytische Auseinandersetzung mit dem konkreten Inhalt der Trainingsfotos vorgenommen. Diese ergab unter anderem, dass sich im Trainingsdatensatz eine Vielzahl von Nahaufnahmen des 3D-Modells befinden, die zum einen aufgrund der begrenzten Auflösung der Texturen des Modells äußerst unscharf erscheinen und zum anderen so nah aufgenommen wurden, dass auf den Bildern keine verwertbaren Informationen zu erkennen sind. Zudem erscheint die Anzahl der Fotos, auf denen sich Ansammlungen von Distraktoren gebildet und somit das gesamte Modell verdeckt haben, relativ groß (Abbildung 4.8a). Zwar gilt weiterhin die Annahme, dass ein mit Deep Learning Verfahren trainiertes Netz durchaus robust gegenüber diesen Störfaktoren sein sollte – was sich letztendlich auch im Training wiederspiegelt (siehe folgender Abschnitt) – doch lässt sich aufgrund der hohen Datenmenge schwer eine Einschätzung darüber treffen, welche Menge von Störfaktoren sich tatsächlich störend auswirkt.

Der für dieses Experiment letzte Datensatz wurde folglich mit der Motivation erstellt, eine realistischere Darstellung innerhalb dieses funktionalen Realismus zu erzeugen (Abbildung 4.8b). Die Kameraführung wurde optimiert, sodass keine Fotos mehr nur den äußeren Rand der Kathedrale oder extreme Nahaufnahmen zeigten. Zudem wurde die Menge der Distraktoren aufgeteilt in zwei Gruppen. Eine Gruppe mit wenigen großen,

(a) Ungeeignet (aus *M-S4*)



(b) Geeignet (aus *M-S5*)

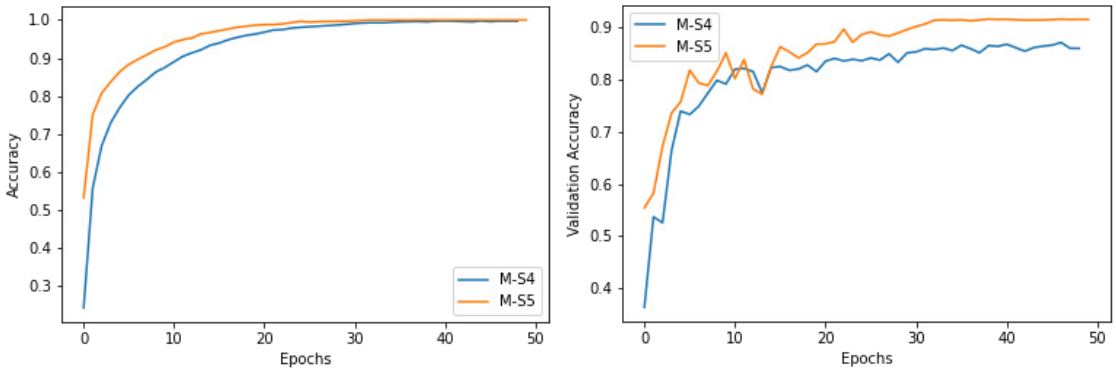
Abbildung 4.8: Beispiele für ungeeignete und geeignete Trainingsfotos

sich langsam bewegenden Distraktoren sollten Störobjekte wie Bäume oder Autos simulieren und eine weitere Gruppe mit vielen kleinen, sich schneller bewegender Distraktoren stellten Menschenmengen auf den begehbaren Plätzen rund um das Gebäude dar. Die Beleuchtungsstrategie wurde durch die Hinzunahme verschiedener Farbtöne und Helligkeiten so angepasst, dass die Texturen in einigen Szenen kräftiger wirken. Die bisherige Beleuchtung sorgte in vielen Fällen für Reflexionen auf den Texturen, wodurch die mangelnde Tiefe des 3D-Modells offenbart wurde. Dies konnte durch den Einsatz schwächeren Lichts mit warmen Farben behoben werden.

Bei der Entwicklung von *M-S5* wurde der Testdatensatz als Referenz genutzt und bei der Erstellung der Daten auf eine gute Abdeckung der Klassen des Testdatensatzes geachtet und der Umkreis darauf begrenzt. So entstand ein Datensatz mit 81 Klassen mit durchschnittlich je 949 bei insgesamt 76.844 Samples.

4.3.2 Training

Für beide Datensätze konnten während des Trainings gute Ergebnisse erzielt werden. Das Modell *M-S4* erreichte eine Trainingsgenauigkeit von 99,78 % und eine Validierungsgenauigkeit von 87,08 % (Abbildung 4.9). Damit wurden zwar nicht die bisher besten



(a) Trainingsgenauigkeit (b) Validierungsgenauigkeit

Abbildung 4.9: Trainingsverläufe des zweiten Durchgangs

Ergebnisse des Datensatzes *M-S3* übertroffen, dafür allerdings eine deutlich hörere Klassenanzahl mit größerem Umkreis abgedeckt. Die Genauigkeit von *M-S5* kann eine etwas steilere Steigung verbuchen, was auf die Datenbereinigung und die geringere Anzahl der Klassen zurückzuführen sein kann und erreicht in Epoche 39 erstmalig 100 %. Auch bei der Validierungsgenauigkeit liegt *M-S5* von Beginn an vorn und zeigt etwa ab Epoche 30 generell ein leicht stetigeres Verhalten. Mit 91,58 % kommt das Modell der Trainingsgenauigkeit somit sehr nahe, sodass das Modell in der synthetischen Domäne eine gute Erkennungsrate liefert (Tabelle 4.4).

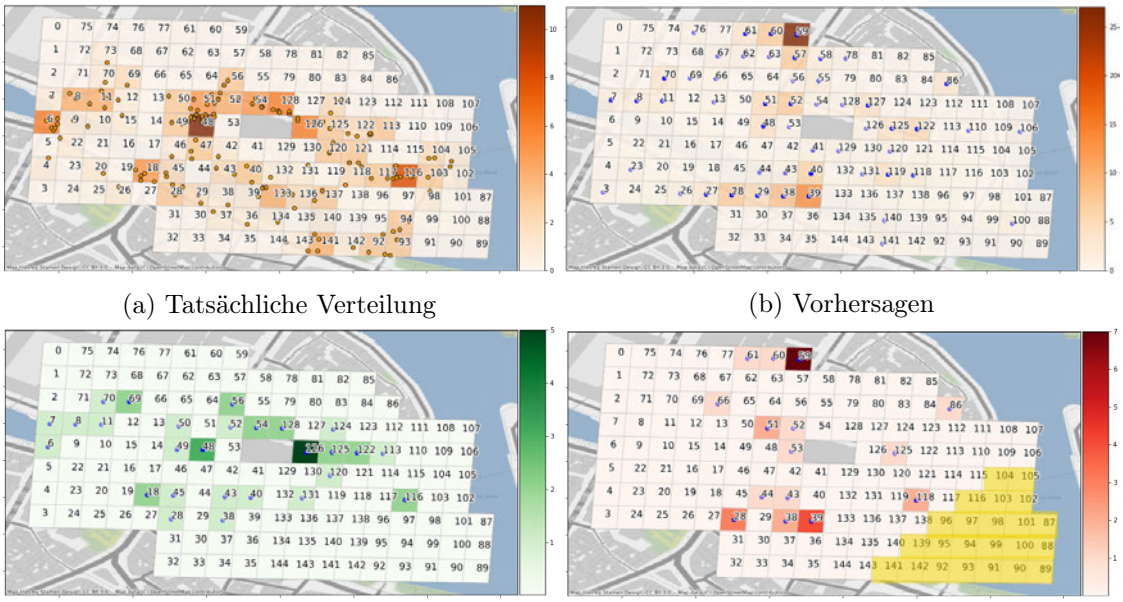
Modell-Bezeichnung	**Eigenschaften**	**Gesamt-Samples**	**Anz. Klassen (min. Fotos / max. Fotos)**	**Max. Acc.**	**Max. Val.-Acc.**	**Epochen**
M-S4	perfekt ausbalanciert	115015	145 (600/800)	0,9978	0,8708	50
M-S5	bereinigter Bildinhalt	76844	81 (400/1000)	1	0,9158	50

Tabelle 4.4: Ergebnisse der Trainings aus dem zweiten Durchgang

4.3.3 Testing

Modell M-S4

Dem Test für das Modell *M-S4* standen 164 Fotos verteilt auf 64 Klassen zur Verfügung. 23 Fotos wurden innerhalb des Umkreises von 50 m von der gesuchten Geoposition gefunden, von denen sieben exakt bestimmt wurden. Mit 14,02 % bzw. 4,27 % Trefferquote liegen die Ergebnisse damit zwischen denen der M1-S1-512 und *M-S1-512s* Modelle, sodass



(a) Tatsächliche Verteilung

(b) Vorhersagen

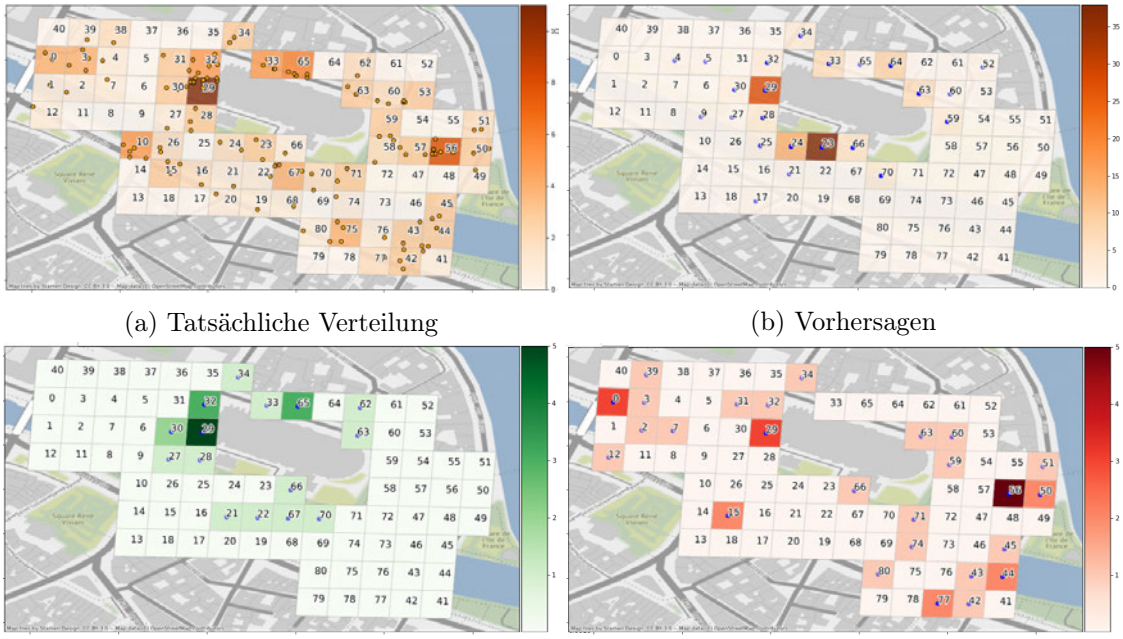
(c) Nur korrekte Vorhersagen

(d) Vorhersagen der markierten Zellen

Abbildung 4.10: Vorhersagen der Testfotos mit Modell *M-S4*

das Ergebnis in Anbetracht der perfekt ausbalancierten Verteilung der Fotos auf die Klassen und der großen Datenmenge unter den Erwartungen bleibt. Auf Klassenebene wird mit 20 Treffern (31,25 %), von denen sechs (9,38 %) exakt getroffen wurden, die Ergebnisse von *M-S1-512s* nahezu erreicht. Generell zeichnet sich bei diesen beiden Modellen ein sehr ähnliches Bild ab. Auch bei *M-S4* gibt es eine extreme Anhäufung in Klasse 59 (Abbildung 4.10b), deren Ursprünge denen aus *M-S1-512s* (Abbildung 4.6d) stark ähneln. Vergleicht man alle getroffenen Vorhersagen dieser beiden Modelle miteinander, kann eine Verschiebung von vielen Vorhersagen im östlichen Bereich zum westlichen Bereich erkannt werden. Dennoch finden sich im östlichen Bereich mehr korrekte Vorhersagen wieder als im Falle des *M-S1-512* Modells. Abbildung 4.10d zeigt die falschen Vorhersagen der gesuchten südöstlichen Zellen zusammengefasst. Durch die weite Verteilung wird die Unsicherheit, mit der das Netz die Einschätzungen trifft, sichtbar. Dies wäre eine von vielen denkbaren Auswirkungen der Vorhersage-Wahrscheinlichkeit, die bei diesem Modell bei nur durchschnittlich 51,65 % liegt. Die meisten getippten Klassen befinden sich allerdings in weiter Entfernung zum 3D-Modell, was die Schlussfolgerung nahelegt, dass die unterschiedlichen Seitenansichten des Modells auch hier nicht gut voneinander unterschieden werden.

Modell M-S5



(a) Tatsächliche Verteilung

(b) Vorhersagen

(c) Nur korrekte Vorhersagen

(d) Ursprünge der vorhergesagten Klasse 23

Abbildung 4.11: Vorhersagen der Testfotos mit Modell *M-S5*

Die bisherigen Ergebnisse deuten darauf hin, dass der Inhalt der Trainingsfotos im Gegensatz zur Klassenanzahl und Verteilung der Fotos eine größere Bedeutung hat, als bisher angenommen. Aus dem Grund wurde als letztes Experiment eine Datensatzbereinigung durchgeführt. Die bisher von keinem der Modelle richtig erkannten äußeren Zellen wurden entfernt, sodass für dieses Modell insgesamt 64 Klassen zur Einteilung von 142 Testfotos zur Auswahl standen. Im Toleranzbereich wurden 24 Fotos richtig erkannt, sodass mit 16,90 % das bisherige Favoritenmodell *M-S1-512s* um einen Prozent übertroffen wurde. Auf Klassenebene schneidet dieses Modell allerdings mit 15 Treffern (27,78 %) im Toleranzbereich etwas schlechter als *M-S1-512s* und *M-S4* ab.

Die grafische Betrachtung dieser Ergebnisse zeigt jedoch, dass sich die meisten Vorhersagen im unmittelbaren Umkreis des Gebäudes befinden (Abbildung 4.11b), die sich in großen Teilen sogar mit den dadurch als richtig klassifizierten realen Positionen (Abbildung 4.11c) decken. Damit werden mehr Positionen mit Nahaufnahmen abgedeckt, als durch die Modelle *M-S2-400* und *M-S3*, die speziell für die Erkennung naher Aufnahmen trainiert wurden. Positiv sticht hier Klasse 29 hervor, die mit fünf Treffern das beste Ergebnis liefert und mit den unmittelbaren Nachbarzellen eine Gruppierung guter Ergeb-

nisse bildet. Damit ist diese Gruppe maßgeblich daran beteiligt, dass die durchschnittliche Vorhersage-Wahrscheinlichkeit dieses Modells mit 71,15 % sechs Prozentpunkte über dem bisher besten Wert, der vom Modell *M-S1-512s* erreicht wurde, liegt.

Dieses Ergebnis zeigt, dass das Modell grundsätzlich in der Lage ist, unterschiedliche Himmelsrichtungen bei der Betrachtung des Gebäudes zu unterscheiden. Trotzdem ist auch hier die Praxistauglichkeit sehr eingeschränkt, da ebenso dieses Modell eine deutliche Anhäufung von falschen Vorhersagen gebildet hat. Diese tritt in Zelle 23 auf und ist von der Position her mit der Anhäufung in Modell *M-S1-512s* zu vergleichen. Eine Betrachtung der Ursprünge dieser Vorhersagen (Abbildung 4.11d) lässt aufgrund der hohen Streuung nur Mutmaßungen zu. Das Modell scheint jedoch grobe Schwierigkeiten mit der Bestimmung von (süd)östlichen Fotos zu haben, da diese fast vollständig der Klasse 23 zugeordnet wurden.

4.4 Zusammenfassung

In diesem Kapitel wurden sechs verschiedene Modelle (basierend auf fünf synthetisch erstellten Datensätzen) mit der InceptionV3-Architektur trainiert und anhand eines Testdatensatzes mit realen annotierten Fotos getestet und exemplarisch ausgewertet. Ziel war, die Auswirkung auf Änderungen der dem neuronalen Netz zugrundeliegenden Trainingsdaten experimentell zu überprüfen, um daraus Rückschlüsse auf die Anforderungen an einen geeigneten synthetischen Datensatz für die Bestimmung der Kameraposition aus Fotos ziehen zu können. Während Tests im ersten Durchgang vorrangig darauf abzielten, mit Änderungen der Auflösung, Anzahl und Verteilung der Samples sowie Einzugsgebiet der insgesamt zu trainierenden Klassen, Einfluss auf die Trainingsergebnisse zu nehmen, wurden im zweiten Durchgang Konsequenzen gezogen und mit großer Datenmenge und weiterer Datensatzbereinigung gute Trainingsergebnisse erzielt. Die Ergebnisse der Tests stellten sich allerdings als durchwachsen heraus, deren Analyse in vielen Fällen vorerst nur auf Mutmaßungen basieren kann. Zur Übersicht sind alle Modelle noch einmal in Tabelle 4.5 zusammengefasst zu sehen.

Grundsätzlich konnten die Ergebnisse aus dem vorausgegangenen Projekt [31] durch diverse Anpassungen in dieser Arbeit deutlich verbessert werden. Ein negativer Effekt ungleicher Verteilung der Samples auf die Klassen konnte nur bei extremer Ungleichheit in einem Ausnahmefall (Modell *M-S3*) beobachtet werden. Ein positiver Effekt im

		Bestimmung der Testfotos			Bestimmung der Testklassen		
Modell-Bezeichnung	**ø Vorhersage-Wahrscheinlichkeit**	**Anzahl**	**Exakte Treffer: relativ / absolut**	**Distanz < 50 m: relativ / absolut**	**Anzahl**	**Exakte Treffer: relativ / absolut**	**Distanz < 50 m: relativ / absolut**
Synth18.600	47,95 %	164	1,22 % / 2	3,66 % / 6	64	3,12 % / 2	3,12 % / 2
Synth18.600.synced	66,26 %	127	4,72 % / 6	11,81 % / 15	41	9,76 % / 4	19,51 % / 8
M-S1-512	44,36 %	173	1,73 % / 3	13,87 % / 24	70	4,29 % / 3	27,14 % / 19
M-S1-512s	65,20 %	132	7,58 % / 10	15,91 % / 21	43	11,63 % / 5	32,56 % / 14
M-S2-400	56,00 %	82	1,22 % / 1	23,17 % / 19	25	4,00 % / 1	40,00 % / 10
M-S3	56,80 %	79	6,33 % / 5	27,85 % / 22	24	8,33 % / 2	25,00 % / 6
M-S4	51,65 %	164	4,27 % / 7	14,02 % / 23	64	9,38 % / 6	31,25 % / 20
M-S5	71,15 %	142	6,36 % / 9	16,90 % / 24	54	9,26 % / 5	27,78 % / 15

Tabelle 4.5: Testergebnisse beider Durchgänge

Sinne einer absolut ausgeglichenen Verteilung konnte nur bedingt erkannt werden. Vielmehr hatte eine kleine Gesamtanzahl von Klassen bei relativ hoher Sample-Dichte positive Effekte auf die Erkennungsrate und Vorhersage-Wahrscheinlichkeit des Klassifikators. Letztere konnte durch die Bereinigung des Trainingsdatensatzes durch manuelle Eingriffe während der Erstellung optimiert werden. Für eine abschließende Beurteilung der Ergebnisse und der daraus gewonnenen Erkenntnisse sind weitere Untersuchungen nötig, die im folgenden Kapitel „Evaluation“ angesprochen und in Teilen durchgeführt werden.

5 Evaluation

5.1 Allgemein

Die Experimente im vorherigen Kapitel haben gezeigt, dass es grundsätzlich möglich ist, aus einem Training rein synthetischer Daten Merkmale innerhalb einer realen Domäne zu erkennen. Als reale Domäne wurde in diesem Zusammenhang die Notre Dame Kathedrale in Paris gewählt und ein neuronales Netz mit Deep Learning-Verfahren trainiert, um die Kameraposition aus Fotos nur anhand der Pixel des Bildes zu bestimmen.

5.1.1 Trainingsergebnisse

Für die Herstellung von Trainingsdaten werden Fotos mit den jeweiligen Geopositionen in Form von Längen- und Breitengrad-Paaren und zusätzlich eine gute Abdeckung der Anzahl von Beispielen aus den verschiedenen Blickwinkeln des Gebäudes benötigt. Für beide Anforderungen erweist sich die Herstellung synthetischer Daten als weitaus flexiblere Lösung als die Verwendung von Internetfotos, da über den Aufnahmeort, Blickwinkel und Anzahl der erstellten Fotos frei entschieden werden kann. Die Geopositionen und weitere Metadaten wie die Lage und Orientierung der Kamera oder der zu trainierenden Objekte werden automatisch exportiert, sodass im Anschluss aus diesen Daten die benötigten Informationen entnommen und für das Training angepasst werden können. Die Problematiken, die sich während der Beschaffung realer Trainingsdaten ergeben haben (nicht oder falsch annotierte Fotos, ungleiche Verteilung der Fotos etc.) treffen bei der Synthese nicht zu und durch die Kontrolle der Erstellung kann ein perfekt ausbalancierter Datensatz mit hoher Abdeckung aller Perspektiven erstellt werden. Die erwähnten Vorteile zeigten sich in dieser Arbeit während des Trainings der Modelle, bei dem eine Trainingsgenauigkeit von 100 % und bei der Validierung rund 92 % erreicht werden konnten. Die Validierung der realen Daten konnte bei gleicher Klassengröße nur 17 % Genauigkeit erreichen. Dieser Umstand ist allerdings dem starken Rauschen der realen Daten

geschuldet, das sich ebenso in den Validierungsdaten befindet und deshalb das Ergebnis stark verzerrt. Dass die Anzahl der Trainingsdaten und besonders die Reduzierung von Rauschen innerhalb der Daten positiven Einfluss auf die Validierung der Modelle haben, konnte mit der Erstellung großer und letztendlich eines bereinigten Datensatzes gezeigt werden (Abschnitt 4.3.1). Dieser erreicht in *M-S5* (siehe Abschnitt 4.3.1) zudem auch die höchste Vorhersage-Wahrscheinlichkeit während der Tests. Diese Arbeit hat weiterhin auch gezeigt, dass das Design der neuronalen Netzarchitektur eine eher untergeordnete Rolle spielt. Die Deep Learning-Architekturen wie InceptionV3 sind heutzutage so weit entwickelt, dass sie problemlos für unterschiedlichste Domänen verwendet werden können und die Anpassung von Hyperparametern nur noch dann eine wichtige Rolle spielt, wenn eine gute Qualität der verwendeten Trainingsdaten gewährleistet werden kann. Probeweise wurden in diesem Kontext noch weitere Trainings mit unterschiedlichen Lernraten und Batch Sizes durchgeführt. Diese hatte allerdings keine nennenswerten Auswirkungen auf das jeweilige Ergebnis. Ein Austauschen der Architektur (vgl. Abschnitt 2.2.2) würde voraussichtlich ähnliche Ergebnisse erzielen und beim Training eher in Dauer und Speicherbedarf variieren, als dass es die Testergebnisse merklich verändern würde.

5.1.2 Testergebnisse

Die Testergebnisse hängen schließlich maßgeblich von der Qualität des verwendeten Trainingsdatensatzes ab, wie mit den beiden Testdurchgängen mit realen Testfotos (vgl. Tabelle 4.5) gezeigt werden konnte. An dieser Stelle ist darauf hinzuweisen, dass durch das hier vorgestellte Testverfahren aufgrund der begrenzten Testdatenmenge nur eine oberflächige Bewertung der Modelle durchgeführt werden kann und soll deshalb auch nur prototypisch die Machbarkeit dieses Ansatzes demonstrieren. Für eine exakte statistische Analyse würde ein Testdatensatz benötigt, der jede mögliche Klasse mit ausreichend Testfotos gut ausbalanciert abdeckt. Erst dadurch wären Aussagen über die trainierte Qualität jeder einzelnen Klasse möglich, die dann in einem *Classification Report* oder einer *Confusion Matrix* dargestellt werden und mit unterschiedlichen statistischen Metriken zur Leistungsmessung wie Recall und Precision behandelt werden können [62] (vgl. Abschnitt 2.2). Auf diese Darstellungsformen wird in diesem Kontext verzichtet, da sie hier keinen Mehrwert bieten würden.

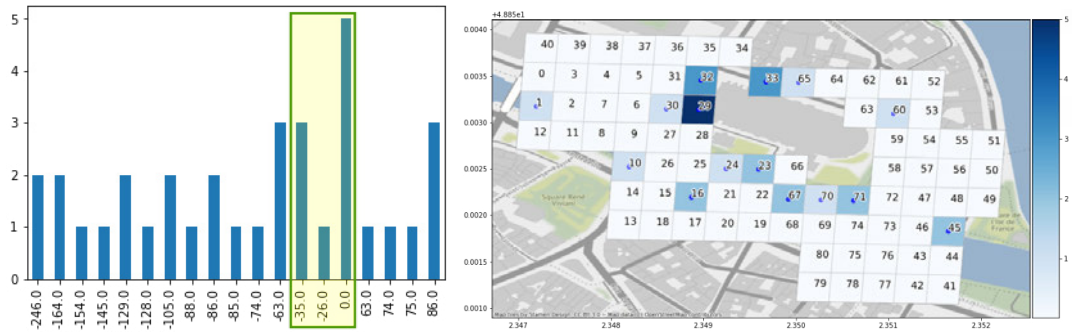
Im Experimente-Teil wurden zudem vordergründig Zahlen für die einzelnen trainierten Modelle erhoben und die jeweilige Vorhersage-Wahrscheinlichkeit mit Datenqualität und Trainingsergebnissen in Verbindung gebracht. Um bessere Schlussfolgerungen aus den

Ergebnissen ziehen und daraus Handlungsoptionen für zukünftige Arbeiten ableiten zu können, ist eine nähere Betrachtung einzelner Vorhersagen nötig. Deswegen werden im Folgenden die Nutzung des Klassifikators und die eingesetzte Metrik zur Beurteilung der Tests diskutiert.

5.2 Nutzung eines Klassifikators

5.2.1 50 m-Metrik

In der Beschreibung des ersten Experiments (Abschnitt 4.2.3) wurde bereits mit dem Abstand zwischen den Mittelpunkten der vorhergesagten und tatsächlich gesuchten Zelle eine weitere Bewertungsgrundlage der Experimente vorgestellt: Statt nur die exakt richtig getroffenen Klassen zu bewerten, werden alle Fotos im Radius von 50 m als akzeptiert betrachtet. Diese oder weitere alternative Betrachtungsweisen sind nötig, da mit dem Klassifikator den Testergebnissen nach keine akzeptable Menge exakter Vorhersagen getroffen werden kann. Um einige mögliche Gründe zu untersuchen, wird anhand eines Beispiels aus dem Test mit Modell *M-S5* im Folgenden veranschaulicht, wie die Nutzung dieser Bewertung in der Praxis aussieht. In Abbildung 5.1a ist ein Histogramm für Zelle 29 zu sehen, das auf der y-Achse die Anzahl und auf der x-Achse die Entfernung der vorhergesagten Zellen vom echten Wert der Fotos aus Zelle 29 in Metern angibt. Die Werte ab null aufwärts sind die Abweichungen der Testfotos aus Zelle 29 (Falsch-Negative), während die negativen Werte durch Fotos, die irrtümlich dieser Zelle zugeordnet wurden (Falsch-Positive), gebildet werden. Die drei grün markierten Balken entsprechen den im Toleranzbereich akzeptierten Vorhersagen, die zur besseren geografischen Einordnung zudem in Abbildung 5.1b durch die Zellen 29, 30 und 32 zu sehen sind. Zelle 29 ist die am häufigsten im Test getroffene Klasse (vgl. Abbildung 4.11c) und doch wird der Großteil der Testfotos entweder falsch vorhergesagt oder irrtümlich für Zelle 29 gehalten.



(a) Histogramm der Verteilung in Metern

(b) Geografische Verteilung der Vorhersagen

Abbildung 5.1: Verteilung der Vorhersagen für Zelle 29 (Modell *M-S5*)

5.2.2 Zellenränder

Ein übliches Problem bei der Verwendung eines Klassifikators für die Bestimmung von kontinuierlichen Werten wie Geopositionen ist die Einordnung der Extrema innerhalb der Klassen. Zwischen benachbarten Zellen können demnach viele Fotos auftreten, die sich sehr ähnlich bis nahezu identisch sehen, aber unterschiedlichen Zellen zugeordnet werden. Die Beispiele in Abbildung 5.2 machen die Problematik besonders am Beispiel der Zelle 29 deutlich: Bilder innerhalb der gleichen Zelle (Vergleich Bild 3 mit Bild 4 v.l.) können deutliche perspektivische Differenzen zeigen, während die Nachbarzelle wiederum eine ähnliche Ansicht enthält (Vergleich Bild 4 mit Bild 5 v.l.).

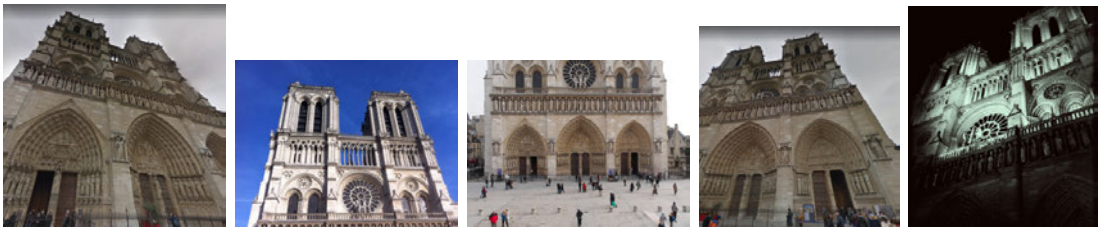


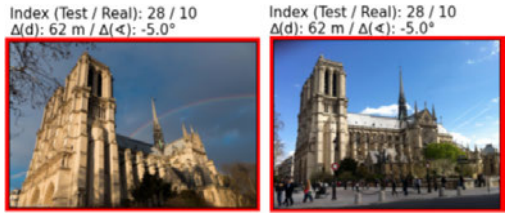
Abbildung 5.2: Fotos aus den Nachbarzellen 32, 31, 29, 29, 30 (v.l.n.r.)

Hinzu kommen Ungenauigkeiten bei der GPS-Bestimmung durch die Sensoren der Quellgeräte oder manuellem Nachtragen und die Nutzung verschiedener Kameraobjektive mit unterschiedlichen Brennweiten oder Zoom-Einstellungen, die ein Verschieben der Fotos über die Zellgrenzen hinweg herbeiführen. Mit der Einführung eines Toleranzbereichs,

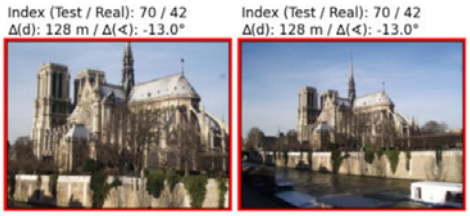
was faktisch einer Gruppierung von Klassen entspricht, wird die Eigenschaft eines Klassifikators, klare Abgrenzungen zwischen den Klassen zu ziehen, umgangen.

5.2.3 Varianz innerhalb einer Zelle

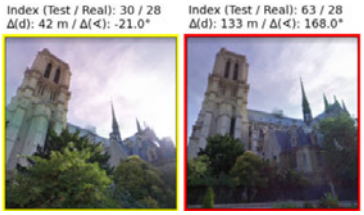
Eine nähere Betrachtung der Fotos innerhalb einer Zelle macht die hohe Varianz der Darstellungen der Notre Dame deutlich. Es finden sich sowohl Detailaufnahmen des Eingangstors oder der Türme wieder als auch Aufnahmen, die den Vorplatz zeigen und zum Teil sogar die gesamte Fassade abdecken können. Eine Übersicht aller Testfotos der Zelle 29 aus *M-S5* mit allen Vorhersagen findet sich im Anhang in Abbildung A.3 wieder und wurde, wie im Hinweis von Abbildung 5.2 beschrieben, annotiert. Die Zelle hat eine Größe von etwa $25 * 36$ Metern. Da sie sich unmittelbar vor dem Gebäude befindet, kann die Kamera sich somit innerhalb der Zelle um einen Winkel von bis zu 27° um den Mittelpunkt des anvisierten Objektes bewegen (vgl. Abbildung 5.4b). Dazu kommen dynamische Rotationen um die eigene Hoch-, Längs- und Querachse im Sinne der sechs Freiheitsgrade, die für unterschiedliche Perspektiven und, wie in Abbildung 5.3b zu sehen, im Test bei kleinen Veränderungen schon für unterschiedliche Vorhersagen sorgen können. Auffällig ist auch, dass Fotos häufig der Zelle 29 zugeordnet werden, wenn eine große Neigung um die Querachse durchgeführt und dadurch ein großer Winkel zwischen Boden und Kirchturmspitze wie im ersten Foto der Abbildung 5.2 aufgespannt wird (vgl. Anhang Abbildung A.3 [Zeile 3, Spalte 4] oder [Z. 4, Sp. 2]). Da dies auch bei Aufnahmen anderer Seiten auftritt, wurde möglicherweise der Fluchtpunkt als Merkmal erkannt.



(a) Bewegung im Winkelbereich von 12°



(c) Zoom innerhalb derselben Klasse



(b) Rotation um die Längsachse innerhalb derselben Klasse

(d) Unterschiedliche Klassen im Winkelbereich der Klasse 42

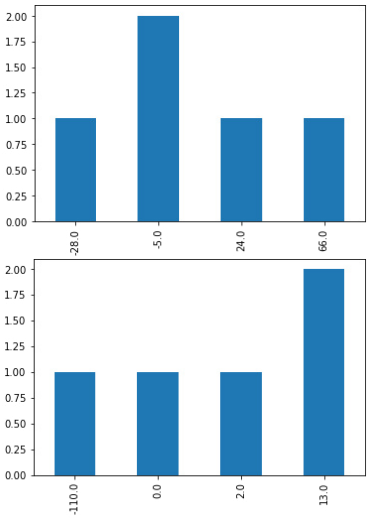
Abbildung 5.3: Rotation und Translation im Testdatensatz[27]

5.2.4 Winkelabweichungen

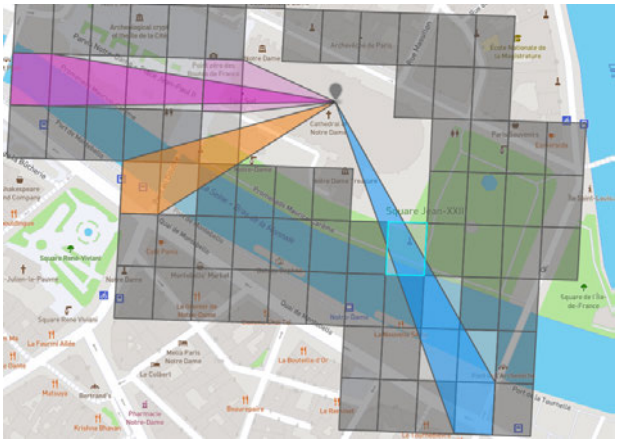
Weitere interessante Erkenntnisse können aus den Fotos gewonnen werden, wenn man die Winkelabweichung der Vorhersagen zum Ursprungsfoto in Relation zum Mittelpunkt der Notre Dame zur Analyse hinzuzieht. Dabei fällt auf, dass die Distanz zwischen Vorhersage und echtem Wert keine ausreichende Grundlage für die Bewertung des Modells bietet. Da die Klassen in statische Größen eingeteilt wurden, ergibt sich das Problem, dass sich der Winkelbereich, in dem sich die Kamera befinden kann, mit geringerer Distanz zum Objekt vergrößert und somit die Varianz innerhalb einer Klasse zunimmt. Für die äußeren Klassen gilt zwar der gleiche Bewegungsspielraum auf der Längs- und Querachse, durch den geringeren Winkel in Relation zum Objekt fällt der Effekt aber deutlich geringer aus. Abbildung 5.4b zeigt die Winkelbereiche für Zelle 29 (kleine magenta Fläche), Zelle 1 (magenta groß), Zelle 10 (orange) und Zelle 42 (gelb). In den nebenstehenden

[27] **Hinweis:** Diese und nachfolgende Fotobeispiele sind zusätzlich mit Testergebnissen versehen worden. Eine grüne Umrandung entspricht einer exakten Vorhersage, gelb einer Erkennung im Toleranzbereich und rot einer falschen Vorhersage. Es werden die Indizes der vorhergesagten und realen Zellen aus Modell *M-S5* angegeben. $\Delta(d)$ ist die Differenz in Metern (negative Werte links vom Ursprung), $\Delta(\measuredangle)$ ist die Winkelabweichung der Kamera relativ zum Mittelpunkt der Notre Dame (negative Werte links vom Ursprung). Auf die Ergebnisse wird nicht immer zwangsläufig eingegangen, sondern sollen bei Interesse zusätzliche Informationen liefern.

Histogrammen (Abbildung 5.4a) sind die Winkelabweichungen gelistet, die im Zusammenhang mit Zelle 10 (oben) und Zelle 70 (unten) vorhergesagt wurden. Bis auf einen Ausreißer, der sich 110° links von der gesuchten Klasse befindet, befinden sich die anderen im in Abbildung 5.4b gelb markierten Bereich. Die entsprechenden Fotos sind in den Abbildungen 5.3c und 5.3d abgebildet. Ein Foto wurde tatsächlich exakt getroffen, die anderen befinden sich 73 m bzw. 128 m vom Ziel entfernt und fallen beide nicht mehr in den Toleranzbereich. Zum einen wird dadurch deutlich, dass das trainierte Modell Probleme mit der Einschätzung der Entfernung der Kamera zum Objekt hat. Weiterhin zeigt es aber auch, dass allgemein mit dieser Vorgehensweise kein Kamerazoom erkannt werden kann, da die Fotos in Abb. 5.3c z.B. innerhalb derselben Klasse fotografiert wurden. Die beiden Fotos in Abb. 5.3a sind ähnliche Beispiele und wurden beide fälschlicherweise in Klasse 28 (Abb. 5.3b) eingeordnet. Die Winkeldifferenz beträgt allerdings nur 5°, sodass beide Fotos im orange markierten Bereich aus Abb. 5.4b liegen. Hier könnte demnach eine Verbesserung der Erfolgsmetrik durch eine sinnvolle Kombination aus Entfernung und Winkeldistanz zwischen Vorhersage und tatsächlichem Wert der Testfotos erfolgen. Dafür wäre natürlich eine geeignete Größe des Testdatensatzes nötig, um aussagekräftige Histogramme erstellen zu können. Die hier gezeigten Beispiele deuten aber schon darauf hin, dass das Netz in Teilen Merkmale gelernt und in passenden Umgebungen wiedererkannt hat.



(a) Winkelabweichung in Zelle 10 (oben) und Zelle 70 (unten)



(b) Winkelbereiche der Zellen 1 und 29 (magenta), 10 (orange) und 42 (blau) mit 70 (hellblau) im Bereich von 42

Abbildung 5.4: Analyse von Winkelabweichungen

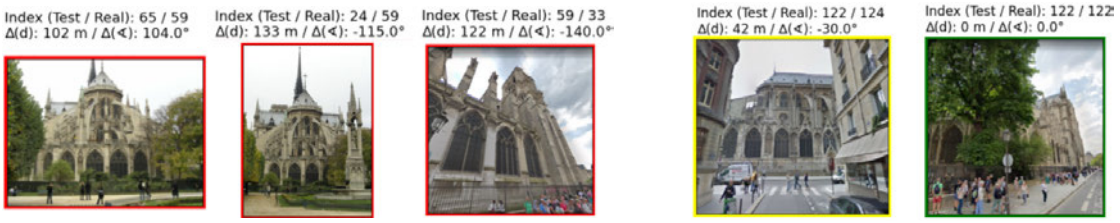
5.3 Bildinformationen

5.3.1 Realitätslücke

Soeben wurden einige positive Testfälle vorgestellt bzw. Verbesserungsmöglichkeiten der Bewertung diskutiert, die mehr positive Ergebnisse akzeptieren würde. Insgesamt treffen alle trainierten Modelle allerdings sehr viele Fehlentscheidungen, die nur schwer und in vielen Fällen nur mit empirischen Tests zu begründen sind. Es gibt aber Tendenzen, die auf bestimmte Verhaltensweisen hindeuten. In einigen Fällen scheinen die Modelle die betrachtete Seite des Notre Dame Modells zu verwechseln. Hinweise darauf gibt es im Falle des Modells *M-S1-512*, mit dem die Ursprünge der falsch klassifizierten Fotos in Zelle 79 untersucht wurden (siehe Abbildung 4.5d). Dies trifft sowohl auf weit entfernte Fotos zu (die im Allgemeinen eine geringere Trefferquote haben), als auch auf Nahaufnahmen in den Modellen *M-S2* und *M-S3*. Da letztere aber entweder mit zu wenigen (*M-S2*) oder ungleich verteilten (*M-S3*) Trainingsfotos trainiert wurden, sollen diese nicht weiter beachtet werden, da das wahrscheinlich diesen Umständen zuzuschreiben ist. Die Probleme, Nord- und Südseite zu unterscheiden, treten allerdings auch im Modell *M-S4* trotz vieler Trainingsfotos und guten Trainingsergebnissen auf. Besonders die mangelnde Erkennung bei größerer Entfernung der Kamera zum Objekt deutet vermutlich auf die Grenzen des Machbaren mit der Domain Randomization-Methode hin. Ein Gebäude wie die Notre Dame besitzt äußerst komplexe Erscheinungsmuster, deren Eigenschaften aus weiter Entfernung wahrscheinlich von den Feature Extraktoren nicht mehr erkannt werden können. Da mit der Domain Randomization das Lernverhalten aber lediglich auf das 3D-Modell reduziert wurde, entsteht hier aufgrund mangelnder weiterer Anhaltspunkte in den Bildern eine Realitätslücke (vgl. Abschnitt 2.5.2), die schlichtweg zu groß für eine ordnungsgemäße Erkennungsrate ist. Abhilfe könnte eine Erweiterung des Trainingsdatensatzes durch die zusätzliche Erstellung fotorealistischer Umgebungsfotos oder korrekt annotierter realer Fotos geschaffen werden, die dem neuronalen Netz geografische Orientierungspunkte z. B. durch charakteristische Umgebungen wie umliegende Gebäude, Straßenlampen, Flüsse und Bäume oder die Umzäunung des Gebäudes liefern. Damit wäre dann allerdings der Anspruch einer einfachen und ressourcenschonenden Erstellung des Datensatzes (vgl. Abschnitt 2.4.1) nicht mehr gegeben.

5.3.2 3D-Modell

Bei der genaueren Analyse einzelner Klassen des letzten Modells *M-S5* konnten einige Verwechselungen zwischen Westfassade (Umgebung der Zelle 29) und Nordfassade (Umgebung der Zelle 33) und besonders zwischen Südfassade (Umgebung der Zelle 23) und Westfassade entdeckt werden. Zwar ist Zelle 23 bei diesem Modell die Anhäufung, die die meisten falsch getroffenen Vorhersagen enthält. Auf vielen Fotos ist aber die charakteristische Fensterrosette deutlich zu erkennen, die der der Westfassade zum Verwechseln ähnlich sieht. Diese ist auch auf der Nordseite zu finden (vgl. Anhang Abb. A.3 [Z. 4, Sp. 2]), wodurch hier Verwechslung nachvollziehbar wirkt. Die in Abschnitt 3.2.4 im Kapitel „Datenmaterial angesprochene Ungenauigkeit des genutzten 3D-Modells in Bezug auf das reale Objekt kommt hier ebenfalls zu tragen, indem das Fehlen der kleinen Kapelle auf der Südseite und der von der Nordseite kopierte Haupt-Seiteneingang bei der Erkennung ungewollte Symmetrie mit der Nordseite erzeugt (vgl. Abbildung 3.6). Die vielen im 3D-Modell vereinfacht dargestellten Formen mit weniger Verschnörkelungen und spitz zulaufenden Formen mitunter an den Eingangstüren, sorgen besonders bei Detailaufnahmen für eine Erhöhung der Diskrepanz zwischen Simulation und Realität. Die Schwierigkeiten bei der Erkennung der rückseitigen Testfotos, die aus dem Garten im Hinterhof stammen, könnten darin begründet sein, dass der Autor des 3D-Modells die untere Fensterreihe für das gesamte Modell verwendet hat (vgl. Anhang Abb. A.2). In der Realität haben die Fenster auf der Rückseite allerdings etwa die doppelte Breite.



(a) Klasse 59 in Modell *M-S5*

(b) Klasse 122 in Modell *M-S4*

Abbildung 5.5: Vorhersagen der Notre Dame Rückseite

Im Test des Modells *M-S5* wurde keins dieser Fotos richtig erkannt. Stattdessen fällt die Vorhersage für das linke Foto in Abbildung 5.5a in den nördlichen Bereich. Die geschätzte Position (Zelle 65) befindet sich unmittelbar neben der Zelle 33, dessen Foto (Abb. 5.5a rechts) wiederum für Klasse 59 (und somit den Bereich des Gartens) gehalten wurde. Hier kann also davon ausgegangen werden, dass das Netz die Form der Fenster gelernt,

diese aber nicht korrekt in den Kontext eingeordnet hat. Das Motiv in Abb. 5.5a (Mitte) wiederum entspricht einer Translation auf der Längsachse verglichen mit dem linken Foto. Die vorhergesagte Zelle 24 liegt allerdings exakt auf der gegenüberliegenden Position der Schätzung des linken Bildes, entspricht also einer Spiegelung des 3D-Modells. Ob der Auslöser dafür wieder die Fenster waren oder der Obelisk auf der rechten Seite im Bild darauf Einfluss genommen hat, kann auf diese Weise nicht bestimmt werden. Eine bessere Erkennung kann hingegen exemplarisch auf der Nordseite entdeckt werden. Die in Abb. 5.5b abgebildeten Fotos wurden durch Modell *M-S4* erkannt und bieten damit überraschend gute Treffer. Warum *M-S4* allerdings im Allgemeinen bessere Ergebnisse in diesem Klassenbereich als *M-S4* erzielen konnte, konnte in der Analyse dieser Arbeit nicht erörtert werden, da sich die Trainingsdatensätze der beiden Modelle in diesem Gebiet sehr ähneln. Um diesen Problemen letztendlich entgegenzuwirken, wäre an dieser Stelle die Nutzung eines geeigneteren 3D-Modells nötig, das die Realität realistisch abbildet und vorzugsweise mit vielen Polygonen (*High Poly Modell*) modelliert wurde, sodass Reflexionen und Schattenwürfe aufgrund zusätzlicher Tiefeninformationen realistischer gerendert werden können. In dem Kontext sollte ferner auf hochauflösende Texturen gesetzt werden, um die Auflösung der Trainingsfotos zu erhöhen und dadurch bessere Ergebnisse bei der Schätzung der Kamerapositionen weit entfernter Fotos erzielen zu können.

5.3.3 Interpretierbarkeit und Anwendbarkeit

Die im Experimente-Teil und in diesem Kapitel gezogenen Rückschlüsse basieren größtenteils auf Mutmaßungen. Die Verkettung vieler nicht-linearer Funktionen in vielschichtigen Deep Learning-Architekturen macht sie hochgradig intransparent und bieten diesbezüglich keine gute wissenschaftliche Fundierung. Die Eingabewerte werden in eine Art „Black Box“ gegeben, die keine Rückschlüsse darüber zulässt, welche Merkmale der Eingabe ausschlaggebend für die Entscheidungsfindung waren [55]. Erklärbarkeit von maschinellem Lernen ist nicht Gegenstand dieser Arbeit, allerdings gibt es in diesem Bereich einige wissenschaftliche Ansätze, die diese Frage untersuchen. Molnar [43] bietet hier einen guten allgemeinen Überblick. Speziell im Bereich der Bilddeutung beschäftigt sich an der HAW Hamburg z. B. Juri Zach [81] im Rahmen seiner Masterarbeit mit dem praktischen Einsatz von verschiedenen Interpretationsmethoden von neuronalen Netzen. An dieser Stelle könnten Untersuchungen unternommen werden, um die Erkennung in diesem

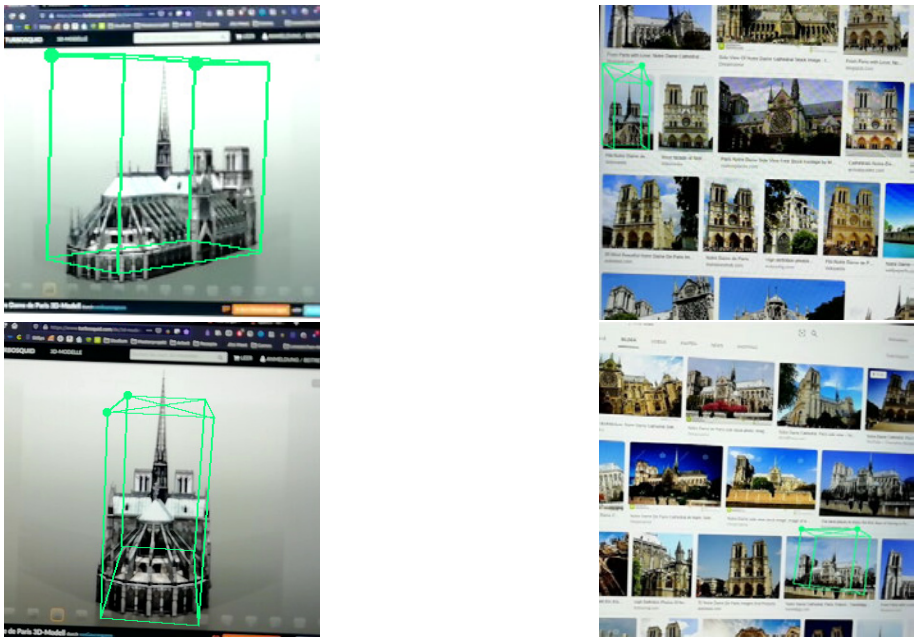
Projekt nachvollziehen zu können und dadurch den Optimierungsprozess durch gezielte Maßnahmen zu beschleunigen.

Während der Testphase und Analyse der realen Daten mit den synthetisch trainierten Modellen sind ebenso Fälle aufgetreten, in denen falsch markierte reale Fotos vom Modell intuitiv richtig eingeordnet wurden. Dies lässt die Annahme zu, dass ein Datensatz ungelabelter oder wie in diesem Fall sehr verrauschter Fotos der Notre Dame mit diesen Modellen zumindest in Teilen verbessert werden können. Dafür wird dann allerdings voraussichtlich eine Erweiterung des Trainingsdatensatzes um weitere Kameraperspektiven (u. a. mit Bewegung auf der z-Achse) nötig sein, um auch professionelle Fotos aus besonderen Perspektiven und/oder Luftaufnahmen abdecken zu können. Nichtsdestotrotz sind die trainierten Modelle auf die Notre Dame als einziges Objekt zugeschnitten. Sollen weitere Objekte erkannt werden und besonders in Anbetracht der ursprünglichen Fragestellung, ob sich diese Vorgehensweise allgemein zum Bereinigen von EXIF-Metadaten in Fotos eignet, sind enorme Datenmengen, rechenintensive Trainings und mitunter individuelle Anpassungen an die jeweiligen Umgebungen nötig. Nach diesem Prinzip wäre also nur eine Erkennung von bekannten Umgebungen möglich, für die gute 3D-Modelle zur Verfügung stehen. Alternativ müssten diese manuell modelliert werden, womit hoher zeitlicher Aufwand und Expertise erforderlich wären. Kombiniert mit einer Trainingsdauer, die trotz der Verwendung von Hochleistungshardware an der HAW Hamburg (vgl. [44]) für ein Training der Modelle *M-S4* oder *M-S5* etwa zwei Tage in Anspruch nahmen, sollte das Kosten-Nutzen-Verhältnis dieses Ansatzes hinterfragt werden.

5.4 Exkurs: Die Notre Dame im DOPE-System

Motiviert durch die sich in der Evaluation ergebene Frage, ob für die Bestimmung von Geokoordinaten ein Multi-Klassen Klassifikator eine gute Wahl ist, wurde als Exkurs ein Versuch zur Erkennung der Notre Dame mit dem ursprünglichen Einsatzzweck des DOPE-Systems [74], das im Kontext des NDDS-Plugins in Abschnitt 2.5 vorgestellt wurde, durchgeführt. In diesem Ansatz werden die sechs Freiheitsgrade von Objekten der Größe haushaltsüblicher Gegenstände trainiert, indem sie um alle Achsen rotiert werden, während die Kamera auf einem festen Punkt mit einem virtuellen Abstand von 0,5 bis 4 Metern fixiert ist und Momentaufnahmen anfertigt. Um in die Einstellung dieser Größenordnung zu passen, musste das 3D-Modell der Notre Dame mit dem Faktor 145 auf diesen Maßstab herunterskaliert in die Umgebung eingesetzt werden. Zusätzlich wurde

eine Cracker Box aus dem *Falling Things*-Datensatz [73], die auch für Trainings in [72] und [63] verwendet wurde, platziert. Diese wurde ebenso in den Datensatz integriert und trainiert, um später bei der Erkennung als Referenzwert sicherzustellen, dass das Modell reibungslos funktioniert. Für die Notre Dame mussten zudem Einschränkungen in der Beweglichkeit vorgenommen werden, um Aufnahmen unrealistischer Perspektiven, die in der Realität nicht entstehen können, zu verhindern. Das 3D-Modell besitzt z. B. keinen modellierten Boden, sodass es nicht vollständig als 6-DoF-Objekt behandelt werden kann (vgl. Problematik in Abschnitt 3.2.2). Folglich wurde ein Datensatz aus 120.000 Fotos, der beide Objekte enthält, erstellt. An den Parametereinstellungen für das Training wurden keine Änderungen vorgenommen und somit ein Training sowohl für das Notre Dame Modell als auch die Cracker Box von je 60 Epochen durchgeführt. Als CNN-Architektur kommt in DOPE das PoseCNN [79] zum Einsatz, das Objekte durch semantische Annotationen klassifizieren und anschließend 3D-Translation und 3D-Rotation per Regression bestimmen kann.



(a) Fotos des trainierten 3D-Modells (b) Ergebnisse einer Suchmaschine

Abbildung 5.6: Testen des trainierten DOPE-Modells auf einem PC-Monitor

Da das DOPE-System im ursprünglichen Sinne für Anwendungen in der Robotik konzipiert wurde, sieht die Anwendung des Systems die Nutzung einer Kamera vor, die stationär die trainierten Gegenstände aufzeichnet und das neuronale Netz währenddes-

sen die Objektposen durch Begrenzungsrahmen markiert. Dabei ist auf den richtigen Abstand der Kamera zu den Objekten zu achten, der dem Kameraabstand in der virtuellen Umgebung entsprechen sollte. In der Praxis würde dies bedeuten, dass das trainierte Notre Dame-Modell in der Realität aufgrund der im Vorfeld durchgeführten Skalierung mit einem Abstand zwischen 100 und 200 Metern die beste Erkennung liefern sollte. Anschließend müsste bei der Berechnung der Pose die Skalierung wieder heruntergerechnet werden. Dieser Praxistest konnte im Rahmen dieser Arbeit nicht durchgeführt werden. Stattdessen wurde eine Smartphone-Kamera als Webcam zur Simulation verwendet, die auf einen PC-Monitor gerichtet war. Auf dem Monitor wurden testweise Bilder des genutzten 3D-Modells zur Überprüfung der Erkennung und Ergebnisse aus einer Bildersuchmaschine mit echten Fotos der Notre Dame angezeigt (siehe Abbildung 5.6). Dieser Schnelltest kam zu dem Ergebnis, dass die Fotos des 3D-Modells (Abb. 5.6a) zwar nur mit einigen Anpassungen der Kameraperspektive und des Abstandes zum Monitor, dann jedoch alle Perspektiven zuverlässig erkannt wurden. Bei den realen Fotos konnte aus einer Sammlung verschiedener Fotos (Abb. 5.6b) in manchen Fällen die richtige Objektpose erkannt werden (in etwa bei einem von 30 Fotos). Dies gelang für verschiedene Posen, allerdings nur dann, wenn das Objekt vollständig zu sehen war und nur bei bestimmter Neigung und Entfernung der Kamera. Tests mit der Cracker Box hingegen wirkten deutlich stabiler. Da das System Vorhersagen nur dann ausgibt, wenn das gesuchte Objekt grundsätzlich gefunden wurde, ist es schwierig, aus dem Verhalten aussagekräftige Schlüsse zu ziehen. Welche genauen Parameter bei der Erkennung eine Rolle spielen, übersteigt zudem den Rahmen dieses Exkurses. Eine ausführliche Messung und Analyse unterschiedlicher Objekt-Kategorien hat Dustin Spallek im Rahmen seiner Masterarbeit [63] an der HAW Hamburg durchgeführt, die an dieser Stelle für weitere Informationen herangezogen werden kann.

Festzuhalten bleibt zudem, dass eine Posenerkennung auch mit Objekten der Größenordnung von Gebäuden mit dem DOPE-System bei der Verwendung von synthetischen Daten als Trainingsdatensatz grundsätzlich möglich ist – auch dann, wenn nur domänenrandomisierte Fotos verwendet werden, die nur dem Grad des funktionalen Realismus entsprechen. Aus einer korrekt erkannten Objektpose und dem Abstand zur Kamera (mit Anpassung des Maßstabes) könnte schließlich die Geoposition berechnet werden. Da die statische Kameraposition und die Umrisse des Gebäudes bekannt sind, würde sich an dieser Stelle eine Überprüfung anbieten, ob damit genauere Bestimmungen der Geopositionen gegenüber des Klassifikator-Ansatzes möglich sind.

5.5 Verbesserungsmöglichkeiten

Tag/Nacht-Modellierung Bisher wurden die Nutzung eines besseren 3D-Modells, Modellierung der Umgebung, Erweiterung des Datensatzes um weitere Kameraperspektiven und die allgemeine weiteren Erhöhung der Auflösung der Trainingsfotos als Möglichkeiten zur Verbesserung der Trainingsergebnisse in Betracht gezogen. In den Tests wurden sich bisher allerdings durch die vorrangige Nutzung von Google Street View Fotos auf Aufnahmen bei Tageslicht bezogen. Um eine allgemeine bessere Erkennungsrate zu gewährleisten und die Realitätslücke weiter zu schließen, wäre eine zusätzliche Modellierung von Nachtaufnahmen möglich, die wiederum weitreichendere Änderungen am Domain Randomization Simulationsaufbau und der Beleuchtungsstrategie sowie Hintergrundauswahl zur Folge hätte.

Segmentierung und Tiefeninformationen In der Analyse wurde beschrieben, dass Annotationen auf das gesamte Bild oder auf einzelne Objekte bezogen sein können. Um einzelne Objekte innerhalb eines Bildes zu klassifizieren, können Segmentierungsverfahren [33] verwendet werden, mit denen die Objekte anhand ihrer Silhouette deutlich vom Hintergrund abgegrenzt werden. Weiter können Tiefeninformationen der auf den Fotos abgebildeten Szenen dem Trainingsdatensatz hinzugefügt werden, indem ein Graustufen-Bild erstellt wird, das Objekte mit zunehmender Entfernung zur Kamera heller repräsentiert. Beispiele sind u. a. im Falling Things-Datensatz [73] zu finden, der im Kontext des DOPE Projektes verwendet wurde. Um diese zusätzlichen Informationen nutzen können, bedarf es einer Erweiterung des verwendeten CNN-Architektur, was somit zum nächsten Punkt, der Wahl der Architektur, führt.

Wahl der Architektur In diesem Ansatz wurde ein simpler CNN-Klassifikator auf Basis der InceptionV3-Architektur verwendet. Simple meint in diesem Kontext, dass die synthetischen Trainingsfotos wie reale Fotos behandelt wurden und auf Bildebene nur mit einer zugewiesenen Zelle annotiert und klassifiziert wurden. Das DOPE-System verwendet hingegen mit PoseCNN ein komplexeres Netz, dass neben einer allgemeinen Klassifikation segmentierter einzelner Objekte mithilfe zusätzlicher Tiefeninformationen die trainierten Objektposen vorhersagen kann. An dieser Stelle wären weitere Versuche möglich, ob mit entsprechend für diesen Anwendungsfall zugeschnittenen Modellen bessere Testergebnisse zu erzielen sind. In dem Zuge könnten zudem Synergieeffekte vortrainierter Netze weiter untersucht werden, indem gezielt nach passenden Domänen gesucht

wird. Babenko et al. [4] konnten z. B. die Erkennung von Landschaften verbessern, indem sie statt eines mit dem ImageNet-Datensatz vortrainierten Netzes ein speziell auf verschiedene Landschaften trainiertes Netz gewählt haben. Möglicherweise eignen sich in diesem Kontext Datensätze, die ausschließlich verschiedene Ansichten von Gebäuden enthalten.

Zurück zu realen Daten Trotz einiger Vorteile der hier verwendeten synthetischen Daten, könnte eine erneute Einbeziehung realer Daten den Testergebnissen zugutekommen. Interessant wäre eine Auseinandersetzung mit der Frage, inwiefern sich synthetische und reale Daten zu einem Datensatz kombinieren lassen. Um einen geeigneten realen Datensatz-Anteil zu erstellen und die Probleme der in Abschnitt 3.1 zu minimieren, wäre die Verwendung unüberwachtem Lernens eine Überlegung, durch das automatische Clustern von Merkmalen, ungeeignete Fotos (wie z. B. Innenaufnahmen) herauszufiltern. Für das Training wäre weiter die Nutzung von *LSTM-Netzen (Long short-term memory)* [26] eine Möglichkeit, Gemeinsamkeiten chronologisch abhängiger Fotos z. B. aus Flickr-Fotoalben zu detektieren und daraus bessere Ergebnisse zu ziehen, was den Autoren des PlaNet-Projektes [78] u. a. gelang.

5.6 Fazit

Die Verwendung synthetischer Trainingsdaten für die Objekterkennung durch neuronale Netze bietet allgemein Potenzial, als Alternative zu realen Daten eingesetzt zu werden. Die kontrollierte Datenerzeugung und die automatische lückenlose Annotation der Objekte vermeidet Rauschen in den Trainingsdaten und ungewollte Anhäufungen bzw. schlecht abgedeckte Bereiche, die zur Über- oder Unterverteilung der Klassen führen. Dadurch entfallen das aufwändige manuelle oder zumindest halb-automatische Nachtragen und Korrigieren der Labels und die Nutzung von Data Augmentation-Verfahren, die bei der Nutzung realer Fotos nötig gewesen wären (vgl. Abschnitt 2.5). Die Domain Randomization-Methode ermöglicht zudem, auch mit niedrigen Hardwareanforderungen effektiv einen großen Datensatz zu erstellen. Zumindest auf der Ebene des Trainings konnten die Vorteile durch gute Ergebnisse von über 90 % Genauigkeit während der Validierung bestätigt werden (Abschnitt 5.1.1). Da diese aber ausschließlich in der simulierten Domäne durchgeführt wurde, ist die Übertragbarkeit in die reale Domäne der

entscheidende Faktor zur Bewertung der Modelle. Der Ansatz dieser Arbeit, die Geopositionen gebündelt in ein statisches Raster einzuteilen, um sie dann durch einen Multi-Class Klassifikator vorhersagen zu lassen, erweist sich generell als funktionsfähig, während er jedoch mit einigen Einschränkungen aufwartet. Die Evaluation der Tests der trainierten Modelle mit dem erstellten Testdatensatz (Abschnitt 4.1.3) ergab schließlich, dass die Granularität der Klassengrößen für eine akzeptable Trefferquote zu fein gewählt wurde. Deshalb wurde mit einer Distanz von unter 50 Metern der Vorhersage zum echten Wert eine weitere Bedingung für die Akzeptanzmenge definiert, womit das Prinzip fester Klassenzugehörigkeiten umgangen wird. Zudem hat die Evaluation mit der Analyse der Winkelabweichungen von Kameraposition in Relation zum Objekt aufgezeigt, dass ein statisches Raster mit Zellen gleicher Größe für die Fragestellung weniger geeignet ist. Durch die zunehmende Auswirkung der Kamerabewegungen innerhalb eines Winkelbereichs bei abnehmender Distanz zum Objekt wäre bspw. eine Kombination der aktuellen Zelleneinteilung mit der eines Tortendiagramms denkbar, um die Daten besser gewichten zu können. Ein weiterer Ansatz ist, die Bestimmung der Geoposition als Regressionsproblem zu sehen und kontinuierliche Werte zu bestimmen. In Form des Exkurses mit dem DOPE-System wurde gezeigt, dass auch dies möglich ist und durch die Bestimmung von Lage und Orientierung des Objektes im Raum eine genauere Berechnung der Geoposition möglich wäre.

Eine genauere Bestimmung der Geokoordinaten hängt in dem vorgestellten Ansatz allerdings in der Hauptsache nicht von der Wahl der Ausgänge des neuronalen Netzes ab, sondern wird maßgeblich durch die Qualität der Trainingsdaten bestimmt. Aufgrund der durchwachsenen Testergebnisse wurde als ausschlaggebendes Kriterium die Realitätslücke identifiziert, die sich bei der Übertragung der synthetischen Domäne in die Realität ergibt. Fehlende Details und zu gering aufgelöste Texturen des eingesetzten 3D-Modells der Notre Dame sowie das Fehlen einer modellierten Umgebung sind Indikatoren für eine Vielzahl der fehlinterpretierten Fotos. Anhand des DOPE-Exkurses konnte zudem durch die bessere Erkennungsrate einer einfachen Cracker-Box der Verdacht erhärtet werden, dass die Notre Dame als Objekt möglicherweise zu komplex für das Vorhaben ist, dem eventuell durch mehr Trainingsdaten oder weiterer Erhöhung der Sample-Auflösung entgegenzutreten wäre.

Am Ende bleibt somit festzuhalten, dass mit dem synthetisch trainierten Klassifikator ein Ansatz vorgestellt wurde, der grundsätzlich funktioniert und unter der Voraussetzung einiger Verbesserungen insbesondere des Datensatzes in bestimmten Anwendungsfällen erfolgreich zur Rekonstruktion der Kameraposition und damit auch zur Korrektur von

EXIF-Informationen in Bezug auf die GPS-Koordinaten eingesetzt werden könnte. Aufgrund der Anwendungsabhängigkeit und der damit verbundenen aufzuwendenden Kosten wie Modellierung und Training, lohnt es sich jedoch, das Kosten-Nutzen-Verhältnis bei der Wahl zwischen realen und synthetischen Trainingsdaten und damit die Praxistauglichkeit zu hinterfragen.

6 Schlussteil

6.1 Zusammenfassung

Diese Arbeit hat sich der Problemstellung gewidmet, die Kameraposition aus Fotos bekannter Objekte mit Deep Learning Verfahren zu rekonstruieren, um Fotos ohne Geoinformationen in den EXIF-Daten nachträglich mit GPS-Koordinaten versehen zu können. Mit dieser Thematik befindet sich die Arbeit im Bereich der Bilderkennung. Zu Beginn wurde dazu im Analyseteil eine thematische Eingliederung durchgeführt, die im Feld der Computer Vision beginnt und analytische Bilderkennungsalgorithmen vorstellt. Über maschinelles Lernen und dem Konzept der tiefen neuronalen Netze im Allgemeinen wurde anschließend eine detailliertere Einführung in das speziell für Bilderkennung optimierte Verfahren der Convolutional Neural Networks (CNN) gegeben.

Mit der InceptionV3-Architektur wurde letztendlich ein von Szegedy et al. [65] entworfenes komplexes Deep Learning Modell vorgestellt, das unter anderem im PlaNet Projekt [78], einem Klassifikator zur pixelbasierten geografischen Bestimmung und Einteilung von Fotos auf der Erde in ein Raster, verwendet wurde. Mit dem NVIDIA Deep Learning Dataset Synthesizer [69] wurde ferner eine Möglichkeit, synthetische Fotos zu erstellen, gefunden. Diese Projekte in Kombination dienten als Grundlage, Fotos in ein geografisches Raster einzuteilen und mit der InceptionV3-Architektur die Erkennung der Geopositionen realer Fotos durchzuführen. Im Gegensatz zum Umfang des PlaNet-Projektes wurde hier lediglich die Erkennung auf Straßenebene durchgeführt und als Domäne ein variabler Umkreis um eine Sehenswürdigkeit ausgewählt.

Kapitel 3 „Datenmaterial“ behandelte die Beschaffung und Erstellung der Trainingsdaten für das neuronale Netz. Als Trainingsobjekt wurde die Notre Dame Kathedrale in Paris ausgewählt, da aufgrund der Bekanntheit eine große Menge verfügbarer Daten zu erwarten war und sie sich zudem aufgrund der besonderen unsymmetrischen Bauweise

gut für die Erkennung von Merkmalen eignet. Somit wurde die Beschaffung realer Fotos aus der Online-Fotoplattform Flickr der Erstellung eigener mit dem NDDS-Plugin synthetisierten Fotos gegenübergestellt und in dem Zusammenhang die Vorgehensweise erläutert, mit der die Annotation der Trainingsdaten mit den Geopositionen und der anschließenden Einteilung in ein Raster erfolgte. Aufgrund starken Rauschens innerhalb der realen Daten in Bezug auf falsch gesetzter Geokoordinaten und unerwarteter Bildmotive, wurde der Entschluss gefasst, in dieser Arbeit das Training mit synthetischen Daten zu optimieren. Diese wurden nach der Domain Randomization-Methode [70] in Form eines funktionalen Realismus erzeugt, um den Modellierungsaufwand und das Rendering bei der Datenerzeugung möglichst gering zu halten.

Der Datenerzeugung folgten Experimente mit unterschiedlichen Datensätzen bzw. Datensatzvariationen, die unter den Gesichtspunkten Trainingsverlauf und Tests mit einem selbst erstellten Testdatensatz realer Fotos beleuchtet wurden. Hier entstand die Erkenntnis, dass Auflösung der Trainingsdaten, Anzahl und Verteilung der Fotos und Klassen sowie die „Reinheit“ der Trainingsfotos zwar Auswirkungen auf die Trainingsergebnisse haben, diese sich aber nicht in der erwarteten Größenordnung in den Tests wiedergefunden haben.

In der Evaluation in Kapitel 5 wurde infolgedessen abschließend eine tiefere Analyse der Testergebnisse mit selbstkritischer Reflexion und einer Anpassung der Bewertungsgrundlage vorgenommen. Hier wurde unter anderem die zu geringe Größe des Testdatensatzes und der dadurch begrenzten Aussagekraft der Testergebnisse bemängelt und die Nutzung eines Klassifikators als Methode zur Bestimmung von Geopositionen der Posenbestimmung mit dem DOPE-System gegenübergestellt. Schlussendlich konnte gezeigt werden, dass die Rekonstruktion der Geopositionen nur anhand der Pixel eines Fotos grundsätzlich auch mit rein synthetisch erstellten Trainingsdaten funktioniert. Trotz der mangelnden Interpretierbarkeit von getroffenen Vorhersagen tiefer neuronaler Netze durch die Testanalyse konnten aber einige Erkenntnisse und Rückschlüsse auf die Verhaltensweise der trainierten Modelle gewonnen werden. In einigen Fällen werden trotz der durchwachsenen Erkennungsrate richtige Klassifizierungen getroffen oder zumindest nachvollziebare Tendenzen erkannt. Unter der Voraussetzung, dass das gezeigte Motiv eines Fotos bekannt ist, könnte dieses Verfahren genutzt werden, um eine genauere Bestimmung der GPS-Koordinaten anzustreben und in einigen Fällen auch die Erkennung in Googles Vision AI (siehe Kapitel 1) zu verbessern.

Als wohl bedeutendste Schwachstellen wurden das bei der Synthese verwendete nicht vollständig detailgetreue 3D-Modell und die nicht modellierte Umgebung der Notre Dame identifiziert. Eine Bestärkung dieser Hypothese erfolgte schließlich zudem durch ein zusätzliches Training des DOPE-Systems zur Bestimmung der Pose von Objekten mit dem Notre Dame Modell. Dieses wurde auf die Größe eines haushaltsüblichen Gegenstandes herunterskaliert und zusammen mit dem 3D-Modell einer Cracker Box trainiert und anschließend die Bestimmung der Pose durchgeführt. Eine Erkennung konnte grundsätzlich erfolgen, die der Cracker Box konnte jedoch weitgehend bessere Ergebnisse erzielen.

Eine verbesserte Erkennungsrate könnte somit mit grundlegenden Änderungen des Trainingsdatensatzes erfolgen. Dabei sollte sich jedoch kritisch mit dem Kosten-Nutzen-Verhältnis auseinandergesetzt werden, besonders in Anbetracht einer generischen Nutzbarkeit des Ansatzes für weitere Objekte.

6.2 Ausblick

Mit der Rekonstruktion der Geoposition aus einem einzigen Foto leistet diese Arbeit einen Beitrag im zurzeit äußerst aktiven Forschungsfeld der visuellen Ortserkennung. Unzählige Arbeiten aus den Bereichen Computer Vision, Machine Learning und Robotik tragen mit diversen Lösungsansätzen dazu bei, Geopositionen, Ortsbeschreibungen und weitere Details bei unterschiedlicher Genauigkeit und Skalierung zu erkennen. Dass das Interesse an dieser Thematik groß ist, liegt mitunter an der Fülle der Anwendungsmöglichkeiten.

Bereinigung und Ergänzung von Metadaten Die Korrektur und Ergänzung von Metadaten wurde in dieser Arbeit anhand des Beispiels von GPS-Koordinaten erprobt. Je nach Vorgehensweise können aber auch intrinsische und extrinsische Kameraparameter bestimmt werden. Mit der Vollständigkeit von Geopositionen in einem Fotoalbum, das z. B. Fotos einer Urlaubsreise enthält, die mit mehreren Geräten aufgenommen wurden, könnten zudem Differenzen der Zeitstempel der verwendeten Kameras gefunden werden, wenn es mehrere Fotos desselben Motivs gibt. Dies kann für die chronologische Sortierung von Urlaubsfotos nützlich sein, indem falsche Zeitzonen korrigiert werden. Die Forschung mit CNNs im Bereich der Bilderkennung geht allerdings viel weiter, indem Hold-Geoffroy et al. [27] z. B. den Sonnenstand aus einem einzigen Foto bestimmen, der vorher mithilfe von 360°-Panoramafotos trainiert wurde. Das Verfahren ermöglicht es, virtuelle 3D-Modelle in realistische Szenen zu platzieren und dabei die Beleuchtung

automatisch korrekt an die Umgebung anzupassen. Hier wäre z. B. interessant zu erforschen, inwieweit eine zeitliche Komponente auf Basis des Sonnenstandes der Bestimmung von Geokoordinaten hinzugefügt werden könnte.

Augmented Reality Anwendungen Ein interessantes Feld bietet die *Augmented Reality (AR)* [29] - die Anreicherung der Realität mit virtuellen Elementen. Die rekonstruierte Kameraposition könnte innerhalb von AR-Anwendungen für mobile Endgeräte oder Dashboard in modernen Autos eingesetzt werden, um ortsgebundene Informationen anzuzeigen. Am Beispiel der Notre Dame könnten z. B. abhängig vom aktuellen Blickwinkel kontextbezogene Informationen und geschichtliche Hintergründe zu der jeweiligen Fassade angezeigt werden.

An dieser Stelle würden sich weitere Untersuchungen bezüglich der Posenschätzung von Objekten anbieten, an die sich im DOPE-Exkurs dieser Arbeit bereits herangetastet wurde. Mit einer exakten Posenbestimmung kann eine AR-Anwendung visuelle Informationen ebenso exakt platzieren. Eine AR-Anwendung könnte um die Möglichkeit erweitert werden, 3D-Modelle visuell in die Realität einzublenden. Damit ließen sich bspw. historische Veränderungen innerhalb einer Stadt einblenden, sodass im Falle der Notre Dame das Gebäude mit der Ansicht vor dem Brand 2019 verglichen werden kann.

Ein weiterer Nutzen vorhergesagter GPS-Koordinaten könnte ferner für autonom gesteuerte Fahrzeuge als zusätzliche Quelle zur eigenen Positionsbestimmung oder der von umgebenden Objekten zur Kollisionsvermeidung und Navigation genutzt werden.

Qualität des Datensatzes Eine große Unbekannte ist durch die Verwendung von Deep Learning Verfahren die Qualität des in dieser Arbeit synthetisch hergestellten Datensatzes. Hier werden Methoden benötigt, mit denen die trainierten Features besser nachzuvollziehen sind, um Aussagen über die Fehleranfälligkeit durch das eingesetzte Low-Poly 3D-Modell zu bestätigen oder falsifizieren. Für eine praxisnahe Anwendung müssen weitere Experimente durchgeführt werden, um u. a. konkrete Aussagen über die Auswirkungen der Verdeckungen treffen zu können. Fragen nach Anzahl und Auswirkungen der Distraktoren, Grad der veränderten Lichtverhältnisse, Bedeutung zusätzlich modellierter Umgebung sind nur einige von vielen, die noch beantwortet werden müssen.

Generalisierbarkeit und Skalierbarkeit Die größten Herausforderungen aller in dieser Arbeit vorgestellten Ansätze sind allerdings Generalisierbarkeit und Skalierbarkeit. Im hier gewählten Verfahren wurden Datensätze für die Erkennung eines einzigen Objektes trainiert. Trotz Hochleistungshardware wurden etwa zwei Tage für den Klassifikations-Ansatz und drei Tage für das Training mit dem DOPE-System benötigt. Zudem wurde hier nur die effiziente Domain Randomization-Methode genutzt, wodurch u. a. die Leistungseinbußen bei der Erkennung zu erklären sind. Zusätzliche Modellierung einer fotorealistischen Umgebung und insbesondere einer gründlichen Modellierung von neuen Objekten würden gerade bei komplexen Objekten im Stile der Notre Dame einen enormen zeitlichen Aufwand und fachliche Expertise erfordern. Hier könnte allerdings zum Teil auf bestehende 3D-Modelle zurückgegriffen und eventuell durch Kooperationen mit CAD-Designern oder Spieleentwicklern entgegengewirkt werden. Weiter müsste für jedes neue Objekt ein neues Training durchgeführt werden, womit die Ansätze für viele Anwendungsfälle nicht praktikabel wären.

Um Geopositionen für möglichst viele Anwendungsfälle in unterschiedlichen Maßstäben bestimmen zu können, kann als weitere Option sowohl die Kombination synthetischer und realer Datensätze als auch die Nutzung unterschiedlicher Herangehensweisen in Kombination untersucht werden. Der Großteil der Arbeiten im Bereich visueller Ortserkennung behandelt die Aufgabe als Image Retrieval Problem, mit dem die Repräsentation eines gegebenen Bildes paarweise mit den Repräsentationen einer großen Bilddatenbank abgeglichen wird. Das Verfahren ist in der Regel zwar sehr rechenintensiv und im Gegensatz zu trainierten Modellen ist für die Bestimmung immer eine Datenbank nötig, die zudem viel Speicherplatz benötigt und deshalb nicht auf Edge Geräten zum Einsatz kommen kann. In der Genauigkeit zeigen sie meist aber sehr gute Ergebnisse und interessante Vorgehensweisen. Torii et al. [71] entwickelten bspw. eine Methode, einen Datensatz mit Google Street View Panoramafotos und den dazugehörigen Tiefenkarten synthetisch zu erweitern, indem sie virtuelle Kameras an verschiedenen Standorten platziert haben, mit denen die gleiche Szene mehrfach aus unterschiedlichen Blickwinkeln aufgenommen wurde. Durch die Vervielfachung des Datensatzes und der Nutzung der Zeitmaschinen-Funktion aus Google Maps konnten sie die Erkennung robuster gegenüber stark variierender Lichtverhältnisse und zeitlichen (optischen) Veränderungen der Objekte machen.

Methoden wie diese oder die Rekonstruktion einer Punkt-Wolke mit Structure-from-Motion-Algorithmen wie in Photo Tourism [61] scheinen für die Rekonstruktion der Kameraposition in Hinblick auf eine generische Anwendung der bessere Ansatz zu sein als

die Verwendung modellierter synthetischer Daten. Eine Kombination aus Klassifikator auf globaler Ebene für die grobe geografische Eingrenzung eines Anfragebildes und der anschließenden Wahl eines geeigneten Datensatzes samt bevorzugtem Verfahren (Image Retrieval, weiterer Klassifikator, Posenbestimmung) für die genaue Bestimmung der Position wäre hier eine denkbare Möglichkeit, die an der Stelle weiter untersucht werden könnte.

Literaturverzeichnis

[1] Afraz, Arash ; Yamins, Daniel L. ; DiCarlo, James J.: Neural mechanisms underlying visual object recognition. In: *Cold Spring Harbor symposia on quantitative biology* Bd. 79 Cold Spring Harbor Laboratory Press (Veranst.), 2014, S. 99–107

[2] Arandjelović, Relja ; Gronat, Petr ; Torii, Akihiko ; Pajdla, Tomas ; Sivic, Josef: *NetVLAD: CNN architecture for weakly supervised place recognition.* 2016

[3] Aubry, M. ; Maturana, D. ; Efros, A. ; Russell, B. ; Sivic, J.: Seeing 3D chairs: exemplar part-based 2D-3D alignment using a large dataset of CAD models. In: *CVPR*, 2014

[4] Babenko, Artem ; Slesarev, Anton ; Chigorin, Alexandr ; Lempitsky, Victor: *Neural Codes for Image Retrieval.* 2014

[5] Bay, Herbert ; Tuytelaars, Tinne ; Van Gool, Luc: SURF: Speeded Up Robust Features. In: Leonardis, Aleš (Hrsg.) ; Bischof, Horst (Hrsg.) ; Pinz, Axel (Hrsg.): *Computer Vision – ECCV 2006.* Berlin, Heidelberg : Springer Berlin Heidelberg, 2006, S. 404–417. – ISBN 978-3-540-33833-8

[6] Brown, Matthew ; Lowe, David G.: Automatic Panoramic Image Stitching Using Invariant Features. In: *Int. J. Comput. Vision* 74 (2007), August, Nr. 1, S. 59–73. – URL https://doi.org/10.1007/s11263-006-0002-3. – ISSN 0920-5691

[7] Cai, Jie ; Luo, Jiawei ; Wang, Shulin ; Yang, Sheng: Feature selection in machine learning: A new perspective. In: *Neurocomputing* 300 (2018), S. 70–79

[8] Cao, Bingyi ; Araujo, André ; Sim, Jack: Unifying Deep Local and Global Features for Image Search. In: Vedaldi, Andrea (Hrsg.) ; Bischof, Horst (Hrsg.) ; Brox, Thomas (Hrsg.) ; Frahm, Jan-Michael (Hrsg.): *Computer Vision – ECCV 2020.* Cham : Springer International Publishing, 2020, S. 726–743. – ISBN 978-3-030-58565-5

[9] Carbonell, Jaime G. ; Michalski, Ryszard S. ; Mitchell, Tom M.: *An Overview of Machine Learning*. S. 3–23. In: Michalski, Ryszard S. (Hrsg.) ; Carbonell, Jaime G. (Hrsg.) ; Mitchell, Tom M. (Hrsg.): *Machine Learning: An Artificial Intelligence Approach*. Berlin, Heidelberg : Springer Berlin Heidelberg, 1983. – URL https://doi.org/10.1007/978-3-662-12405-5_1. – ISBN 978-3-662-12405-5

[10] Chen, C. ; Seff, A. ; Kornhauser, A. ; Xiao, J.: DeepDriving: Learning Affordance for Direct Perception in Autonomous Driving. In: *2015 IEEE International Conference on Computer Vision (ICCV)*, 2015, S. 2722–2730

[11] Chen, L. ; Wang, S. ; Fan, W. ; Sun, J. ; Naoi, S.: Beyond human recognition: A CNN-based framework for handwritten character recognition. In: *2015 3rd IAPR Asian Conference on Pattern Recognition (ACPR)*, 2015, S. 695–699

[12] Cloud, Google: Advanced Guide to Inception v3 on Cloud TPU. (2021). – URL https://cloud.google.com/tpu/docs/inception-v3-advanced. – Zugriffsdatum: 2021-03-21

[13] El Naqa, Issam ; Murphy, Martin J.: *What Is Machine Learning?* S. 3–11. In: El Naqa, Issam (Hrsg.) ; Li, Ruijiang (Hrsg.) ; Murphy, Martin J. (Hrsg.): *Machine Learning in Radiation Oncology: Theory and Applications*. Cham : Springer International Publishing, 2015. – URL https://doi.org/10.1007/978-3-319-18305-3_1. – ISBN 978-3-319-18305-3

[14] Fayyad, Usama: Knowledge discovery in databases: An overview. In: Lavrač, Nada (Hrsg.) ; Džeroski, Sašo (Hrsg.): *Inductive Logic Programming*. Berlin, Heidelberg : Springer Berlin Heidelberg, 1997, S. 1–16. – ISBN 978-3-540-69587-5

[15] Ferwerda, James A.: Three varieties of realism in computer graphics. In: *Human Vision and Electronic Imaging VIII* Bd. 5007 International Society for Optics and Photonics (Veranst.), 2003, S. 290–297

[16] Fischler, M. A. ; Elschlager, R. A.: The Representation and Matching of Pictorial Structures. In: *IEEE Transactions on Computers* C-22 (1973), Nr. 1, S. 67–92

[17] Fossati, Andrea ; Dimitrijevic, Miodrag ; Lepetit, Vincent ; Fua, Pascal: Bridging the Gap between Detection and Tracking for 3D Monocular Video-Based Motion Capture, 07 2007, S. 1 – 8

[18] Frochte, J ö rg: *Maschinelles Lernen: Grundlagen und Algorithmen in Python*. Carl Hanser Verlag GmbH Co KG, 2020

[19] Furukawa, Yasutaka ; ndez, Carlos Hern á: Multi-view stereo: A tutorial. In: *Foundations and Trends ® in Computer Graphics and Vision* 9 (2015), Nr. 1-2, S. 1–148

[20] Gardner, M.W ; Dorling, S.R: Artificial neural networks (the multilayer perceptron)—a review of applications in the atmospheric sciences. In: *Atmospheric Environment* 32 (1998), Nr. 14, S. 2627–2636. – URL https://www.sciencedirect.com/science/article/pii/S1352231097004470. – ISSN 1352-2310

[21] Goesele, Michael ; Snavely, Noah ; Curless, Brian ; Hoppe, Hugues ; Seitz, Steven M.: Multi-view stereo for community photo collections. In: *2007 IEEE 11th International Conference on Computer Vision* IEEE (Veranst.), 2007, S. 1–8

[22] Goodfellow, Ian ; Bengio, Yoshua ; Courville, Aaron: *Deep Learning*. MIT Press, 2016. – http://www.deeplearningbook.org

[23] Hays, J. ; Efros, A. A.: IM2GPS: estimating geographic information from a single image. In: *2008 IEEE Conference on Computer Vision and Pattern Recognition*, June 2008, S. 1–8. – ISSN 1063-6919

[24] Hearst, M. A. ; Dumais, S. T. ; Osuna, E. ; Platt, J. ; Scholkopf, B.: Support vector machines. In: *IEEE Intelligent Systems and their Applications* 13 (1998), Nr. 4, S. 18–28

[25] Hinterstoisser, Stefan ; Lepetit, Vincent ; Wohlhart, Paul ; Konolige, Kurt: On pre-trained image features and synthetic images for deep learning. In: *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*, 2018, S. 0–0

[26] Hochreiter, Sepp ; Schmidhuber, Jürgen: Long short-term memory. In: *Neural computation* 9 (1997), Nr. 8, S. 1735–1780

[27] Hold-Geoffroy, Yannick ; Sunkavalli, Kalyan ; Hadap, Sunil ; Gambaretto, Emiliano ; Lalonde, Jean-Francois: Deep Outdoor Illumination Estimation. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017

[28] Jaderberg, Max ; Simonyan, Karen ; Vedaldi, Andrea ; Zisserman, Andrew: *Synthetic Data and Artificial Neural Networks for Natural Scene Text Recognition*. 2014

[29] Kanne-Schludde, Thomas: Augmented Reality als Unterstützung bei Montage-Vorgängen. (2017). – URL https://users.informatik.haw-hamburg.de/~ubicomp/projekte/master2017-gsem/Kanne/bericht.pdf

[30] Kanne-Schludde, Thomas: Aufbau einer Pipeline zur Bestimmung von GPS-Koordinaten aus Fotos mithilfe neuronaler Netze. (2020), Feb. – URL https://users.informatik.haw-hamburg.de/~ubicomp/projekte/master2020-proj/kanne.pdf

[31] Kanne-Schludde, Thomas: Erzeugung synthetischer Trainingsdaten für die Deep Learning basierte Bestimmung von GPS-Koordinaten aus Fotos am Beispiel der Notre Dame. (2020), Sep. – URL https://users.informatik.haw-hamburg.de/~ubicomp/projekte/master2020-proj/kanne_hp.pdf

[32] Khan, Asifullah ; Sohail, Anabia ; Zahoora, Umme ; Qureshi, Aqsa S.: A survey of the recent architectures of deep convolutional neural networks. In: *Artificial Intelligence Review* (2020), Apr. – URL http://dx.doi.org/10.1007/s10462-020-09825-6. – ISSN 1573-7462

[33] Khan, Waseem: Image segmentation techniques: A survey. In: *Journal of Image and Graphics* 1 (2013), Nr. 4, S. 166–170

[34] Krizhevsky, Alex ; Sutskever, Ilya ; Hinton, Geoffrey E.: ImageNet Classification with Deep Convolutional Neural Networks. In: *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*. Red Hook, NY, USA : Curran Associates Inc., 2012 (NIPS'12), S. 1097–1105

[35] LeCun, Yann u. a.: Generalization and network design strategies. In: *Connectionism in perspective* 19 (1989), S. 143–155

[36] Lin, Tsung-Yi ; Maire, Michael ; Belongie, Serge ; Hays, James ; Perona, Pietro ; Ramanan, Deva ; Dollár, Piotr ; Zitnick, C L.: Microsoft coco: Common objects in context. In: *European conference on computer vision* Springer (Veranst.), 2014, S. 740–755

[37] Liu, Xiaolong ; Deng, Zhidong ; Yang, Yuhan: Recent progress in semantic image segmentation. In: *Artificial Intelligence Review* 52 (2019), Nr. 2, S. 1089–1106

[38] Lowe, David G.: Distinctive image features from scale-invariant keypoints. In: *International journal of computer vision* 60 (2004), Nr. 2, S. 91–110

[39] Lowry, S. ; Sünderhauf, N. ; Newman, P. ; Leonard, J. J. ; Cox, D. ; Corke, P. ; Milford, M. J.: Visual Place Recognition: A Survey. In: *IEEE Transactions on Robotics* 32 (2016), Nr. 1, S. 1–19

[40] Masone, C. ; Caputo, B.: A Survey on Deep Visual Place Recognition. In: *IEEE Access* 9 (2021), S. 19516–19547

[41] Mikolajczyk, K. ; Schmid, C.: A performance evaluation of local descriptors. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 27 (2005), Nr. 10, S. 1615–1630

[42] Mikolajczyk, K. ; Tuytelaars, T. ; Schmid, C. ; Zisserman, A. ; Matas, J. ; Schaffalitzky, F. ; Kadir, T. ; Gool, L. V.: A comparison of affine region detectors. In: *International Journal of Computer Vision* 65 (2005), S. 2005

[43] Molnar, Christoph: *Interpretable Machine Learning*. 2019. – https://christophm.github.io/interpretable-ml-book/

[44] Nitsche, Matthias ; Halbritter, Stephan: Development of an End-to-End Deep Learning Pipeline, Hochschule für Angewandte Wissenschaften Hamburg, 2019

[45] Nixon, Mark ; Aguado, Alberto: *Feature extraction and image processing for computer vision*. Academic press, 2019

[46] Oliva, Aude ; Torralba, Antonio: Building the gist of a scene: The role of global image features in recognition. In: *Progress in brain research* 155 (2006), S. 23–36

[47] Pan, Sinno J. ; Yang, Qiang: A survey on transfer learning. In: *IEEE Transactions on knowledge and data engineering* 22 (2009), Nr. 10, S. 1345–1359

[48] Paul, Richard P.: *Robot manipulators: mathematics, programming, and control: the computer control of robot manipulators*. Richard Paul, 1981

[49] Peng, Xingchao ; Sun, Baochen ; Ali, Karim ; Saenko, Kate: Exploring Invariances in Deep Convolutional Neural Networks Using Synthetic Images. In: *CoRR* abs/1412.7122 (2014). – URL http://arxiv.org/abs/1412.7122

[50] Perez, Luis ; Wang, Jason: The effectiveness of data augmentation in image classification using deep learning. In: *arXiv preprint arXiv:1712.04621* (2017)

[51] Piasco, Nathan ; Sidibé, Désiré ; Demonceaux, Cédric ; Gouet-Brunet, Valérie: A survey on Visual-Based Localization: On the benefit of heterogeneous data. In: *Pattern Recognition* 74 (2018), Februar, S. 90 – 109. – URL https://hal.archives-ouvertes.fr/hal-01744680

[52] Richardson, Elad ; Sela, Matan ; Kimmel, Ron: 3D face reconstruction by learning from synthetic data. In: *2016 fourth international conference on 3D vision (3DV)* IEEE (Veranst.), 2016, S. 460–469

[53] Rosenblatt, Frank: The perceptron: a probabilistic model for information storage and organization in the brain. In: *Psychological review* 65 (1958), Nr. 6, S. 386

[54] Rublee, E. ; Rabaud, V. ; Konolige, K. ; Bradski, G.: ORB: An efficient alternative to SIFT or SURF. In: *2011 International Conference on Computer Vision*, 2011, S. 2564–2571

[55] Samek, Wojciech ; Wiegand, Thomas ; Müller, Klaus-Robert: *Explainable Artificial Intelligence: Understanding, Visualizing and Interpreting Deep Learning Models.* 2017

[56] Samuel, Arthur L.: Some studies in machine learning using the game of checkers. In: *IBM Journal of research and development* 3 (1959), Nr. 3, S. 210–229

[57] Sathya, R ; Abraham, Annamma: Comparison of supervised and unsupervised learning algorithms for pattern classification. In: *International Journal of Advanced Research in Artificial Intelligence* 2 (2013), Nr. 2, S. 34–38

[58] Seitz, S. M. ; Curless, B. ; Diebel, J. ; Scharstein, D. ; Szeliski, R.: A Comparison and Evaluation of Multi-View Stereo Reconstruction Algorithms. In: *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)* Bd. 1, 2006, S. 519–528

[59] Seo, Paul H. ; Weyand, Tobias ; Sim, Jack ; Han, Bohyung: CPlaNet: Enhancing Image Geolocalization by Combinatorial Partitioning of Maps. In: *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018

[60] Shekhar, Ravi ; Jawahar, CV: Word image retrieval using bag of visual words. In: *2012 10th IAPR International Workshop on Document Analysis Systems* IEEE (Veranst.), 2012, S. 297–301

[61] Snavely, Noah ; Seitz, Steven M. ; Szeliski, Richard: Photo Tourism: Exploring Photo Collections in 3D. In: *ACM Trans. Graph.* 25 (2006), Juli, Nr. 3, S. 835–846. – URL https://doi.org/10.1145/1141911.1141964. – ISSN 0730-0301

[62] Sokolova, Marina ; Lapalme, Guy: A systematic analysis of performance measures for classification tasks. In: *Information Processing Management* 45 (2009), Nr. 4, S. 427–437. – URL https://www.sciencedirect.com/science/article/pii/S0306457309000259. – ISSN 0306-4573

[63] Spallek, Dustin: Deep Learning basierte Erkennung von 3D-Objektposen auf Basis synthetisch erzeugter Daten, Hochschule für Angewandte Wissenschaften Hamburg, 2020

[64] Szegedy, Christian ; Liu, Wei ; Jia, Yangqing ; Sermanet, Pierre ; Reed, Scott E. ; Anguelov, Dragomir ; Erhan, Dumitru ; Vanhoucke, Vincent ; Rabinovich, Andrew: Going Deeper with Convolutions. In: *CoRR* abs/1409.4842 (2014). – URL http://arxiv.org/abs/1409.4842

[65] Szegedy, Christian ; Vanhoucke, Vincent ; Ioffe, Sergey ; Shlens, Jonathon ; Wojna, Zbigniew: *Rethinking the Inception Architecture for Computer Vision*. 2015

[66] Tareen, Shaharyar Ahmed K. ; Saleem, Zahra: A comparative analysis of sift, surf, kaze, akaze, orb, and brisk. In: *2018 International conference on computing, mathematics and engineering technologies (iCoMET)* IEEE (Veranst.), 2018, S. 1–10

[67] Taylor, G. R. ; Chosak, A. J. ; Brewer, P. C.: OVVV: Using Virtual Worlds to Design and Evaluate Surveillance Systems. In: *2007 IEEE Conference on Computer Vision and Pattern Recognition*, 2007, S. 1–8

[68] Tesic, J.: Metadata practices for consumer photos. In: *IEEE MultiMedia* 12 (2005), Nr. 3, S. 86–92

[69] To, Thang ; Tremblay, Jonathan ; McKay, Duncan ; Yamaguchi, Yukie ; Leung, Kirby ; Balanon, Adrian ; Cheng, Jia ; Hodge, William ; Birchfield, Stan: *NDDS : NVIDIA Deep Learning Dataset Synthesizer*. 2018. – https://github.com/NVIDIA/Dataset_Synthesizer

[70] Tobin, Josh ; Fong, Rachel ; Ray, Alex ; Schneider, Jonas ; Zaremba, Wojciech ; Abbeel, Pieter: *Domain Randomization for Transferring Deep Neural Networks from Simulation to the Real World*. 2017

[71] Torii, A. ; Arandjelović, R. ; Sivic, J. ; Okutomi, M. ; Pajdla, T.: 24/7 place recognition by view synthesis. In: *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, S. 1808–1817

[72] Tremblay, Jonathan ; Prakash, Aayush ; Acuna, David ; Brophy, Mark ; Jampani, Varun ; Anil, Cem ; To, Thang ; Cameracci, Eric ; Boochoon, Shaad ; Birchfield, Stan: Training Deep Networks with Synthetic Data: Bridging the Reality Gap by Domain Randomization. In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)* (2018), Jun. – URL http://dx.doi.org/10.1109/CVPRW.2018.00143. ISBN 9781538661000

[73] Tremblay, Jonathan ; To, Thang ; Birchfield, Stan: Falling Things: A Synthetic Dataset for 3D Object Detection and Pose Estimation. In: *CoRR* abs/1804.06534 (2018). – URL http://arxiv.org/abs/1804.06534

[74] Tremblay, Jonathan ; To, Thang ; Sundaralingam, Balakumar ; Xiang, Yu ; Fox, Dieter ; Birchfield, Stan: *Deep Object Pose Estimation for Semantic Robotic Grasping of Household Objects*. 2018

[75] Tsuji, Saburo ; Matsumoto, Fumio: Detection of ellipses by a modified Hough transformation. In: *IEEE transactions on computers* (1978), Nr. 8, S. 777–781

[76] Vo, Nam ; Jacobs, Nathan ; Hays, James: Revisiting IM2GPS in the Deep Learning Era. In: *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Oct 2017

[77] VuMa: Anteil der Smartphone-Nutzer in Deutschland in den Jahren 2012 bis 2020 [Graph], Statista GmbH, 2020. – URL https://de.statista.com/statistik/daten/studie/585883/umfrage/anteil-der-smartphone-nutzer-in-deutschland/. – Zugriffsdatum: 2021-03-19

[78] Weyand, Tobias ; Kostrikov, Ilya ; Philbin, James: PlaNet - Photo Geolocation with Convolutional Neural Networks. In: *European Conference on Computer Vision (ECCV)*, 2016

[79] Xiang, Yu ; Schmidt, Tanner ; Narayanan, Venkatraman ; Fox, Dieter: *PoseCNN: A Convolutional Neural Network for 6D Object Pose Estimation in Cluttered Scenes*. 2018

[80] Xuehong Mao ; Hijazi, S. ; Casas, R. ; Kaul, P. ; Kumar, R. ; Rowen, C.: Hierarchical CNN for traffic sign recognition. In: *2016 IEEE Intelligent Vehicles Symposium (IV)*, 2016, S. 130–135

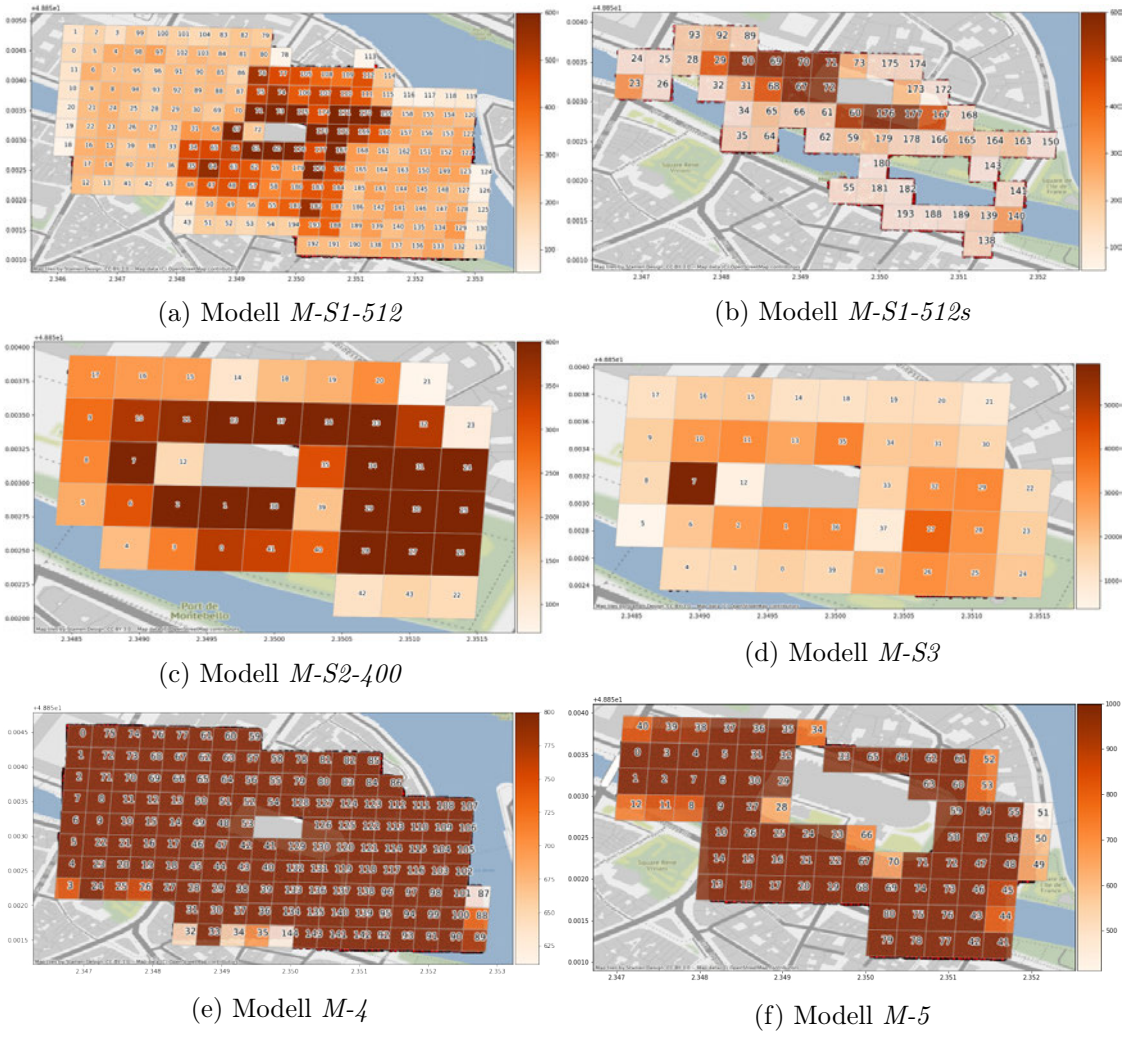
[81] Zach, Juri: Entwicklung einer Softwareumgebung zumInterpretieren von neuronalen Netzen, Hochschule für Angewandte Wissenschaften Hamburg, 2020. – URL https://users.informatik.haw-hamburg.de/~ubicomp/projekte/master2020-proj/zach.pdf

[82] Zamir, Amir R. ; Shah, Mubarak: Accurate Image Localization Based on Google Maps Street View. In: *Proceedings of the 11th European Conference on Computer Vision: Part IV*. Berlin, Heidelberg : Springer-Verlag, 2010 (ECCV'10), S. 255–268. – ISBN 364215560X

[83] Zheng, Alice ; Casari, Amanda: *Feature engineering for machine learning: principles and techniques for data scientists*. Ö'Reilly Media, Inc.", 2018

[84] Zheng, Liang ; Yang, Yi ; Tian, Qi: *SIFT Meets CNN: A Decade Survey of Instance Retrieval*. 2017

A Anhang



(a) Modell *M-S1-512*

(b) Modell *M-S1-512s*

(c) Modell *M-S2-400*

(d) Modell *M-S3*

(e) Modell *M-4*

(f) Modell *M-5*

Abbildung A.1: Verteilung der Training-Samples aller synthetischen Datensätze

Abbildung A.2: Auszug aus dem Trainingsdatensatz für Zelle 122 (Modell *M-S4*)

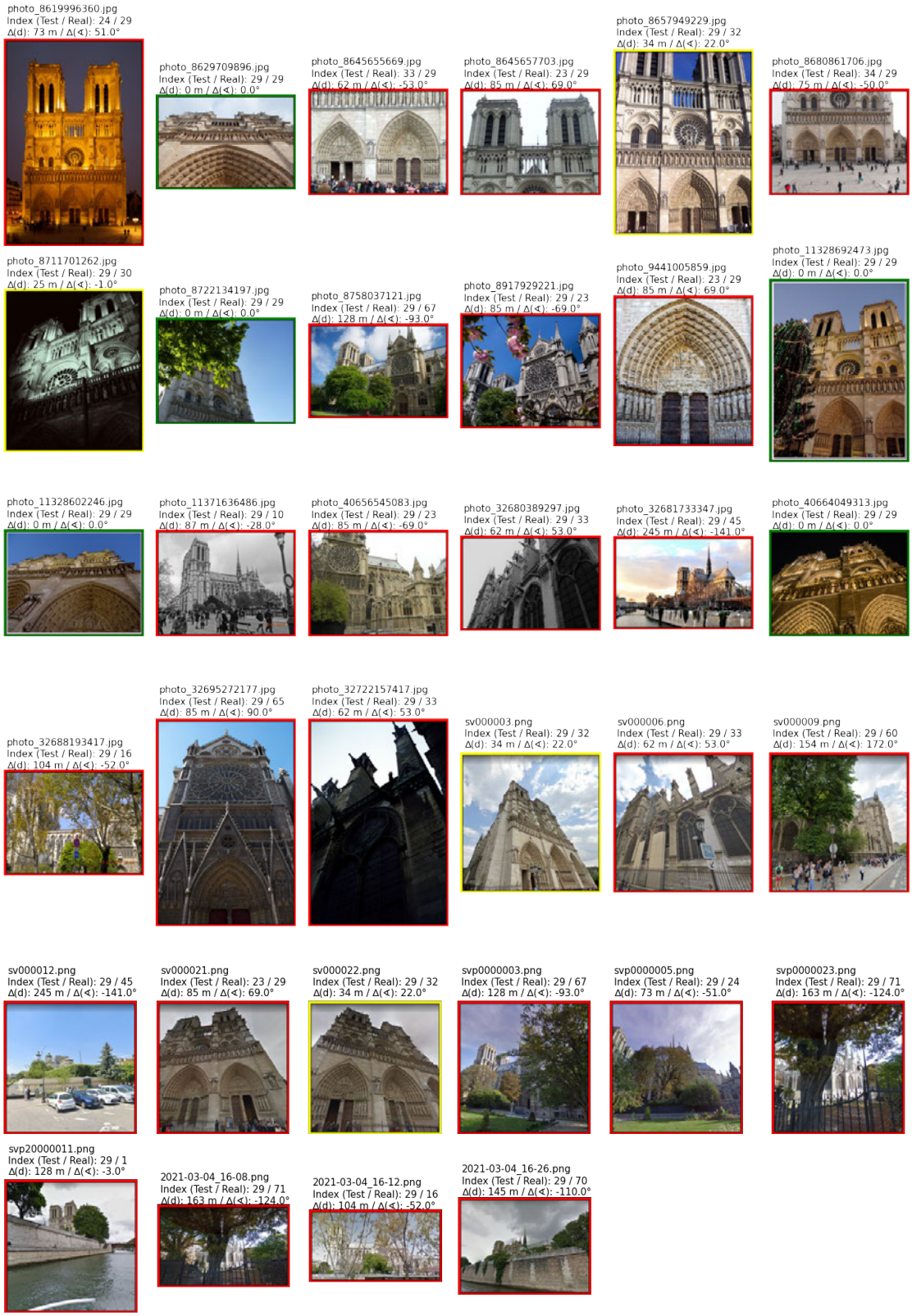


Abbildung A.3: Fotos aus Zelle 29 (Modell *M-S5*) und ihre Vorhersagen. $\Delta(d)$ = Distanzdifferenz und $\Delta(\measuredangle)$ = Winkelabweichung

Erklärung zur selbstständigen Bearbeitung einer Abschlussarbeit

Gemäß der Allgemeinen Prüfungs- und Studienordnung ist zusammen mit der Abschlussarbeit eine schriftliche Erklärung abzugeben, in der der Studierende bestätigt, dass die Abschlussarbeit „— bei einer Gruppenarbeit die entsprechend gekennzeichneten Teile der Arbeit [(§ 18 Abs. 1 APSO-TI-BM bzw. § 21 Abs. 1 APSO-INGI)] — ohne fremde Hilfe selbständig verfasst und nur die angegebenen Quellen und Hilfsmittel benutzt wurden. Wörtlich oder dem Sinn nach aus anderen Werken entnommene Stellen sind unter Angabe der Quellen kenntlich zu machen."

Quelle: § 16 Abs. 5 APSO-TI-BM bzw. § 15 Abs. 6 APSO-INGI

Erklärung zur selbstständigen Bearbeitung der Arbeit

Hiermit versichere ich,

Name: _____

Vorname: _____

dass ich die vorliegende Masterarbeit – bzw. bei einer Gruppenarbeit die entsprechend gekennzeichneten Teile der Arbeit – mit dem Thema:

Rekonstruktion der Kameraposition aus Fotos bekannter Objekte mit Deep Learning-Verfahren

ohne fremde Hilfe selbständig verfasst und nur die angegebenen Quellen und Hilfsmittel benutzt habe. Wörtlich oder dem Sinn nach aus anderen Werken entnommene Stellen sind unter Angabe der Quellen kenntlich gemacht.

Ort	Datum	Unterschrift im Original