Wie deutscher Sonnenschein genutzt wird: Lokalisierung von Solaranlagen mithilfe von Orthophotos und Kartendaten

Simon Petrik und Felix Hamme

1. Juni 2020

Studienarbeit im 5. und 6. Semester an der Dualen Hochschule Baden-Württemberg Karlsruhe

Abgabedatum	1. Juni 2020
Lizenz	Creative Commons Attribution-NonCommercial 4.0 International

Simon Petrik

Ausbildungsfirma SAP SE

Felix Hamme

Ausbildungsfirma United Internet AG

Zusammenfassung

Diese Arbeit beschäftigt sich damit, auf Gebäudedächern angebrachte Solaranlagen anhand von öffentlichen Datenquellen zu lokalisieren. Dazu werden von den Bundesländern veröffentlichte digitale Orthophotos und Kartendaten aus *OpenStreetMap* verwendet. Außerdem werden ein bestehender Datensatz von kalifornischen Orten (vgl. [1]) und Ideen von bisherigen Arbeiten verwendet. Der Stand der Forschung wird ausführlich untersucht.

Die Verfügbarkeit und Qualität von Orthophotos und Kartendaten zu deutschen Regionen für den beschriebenen Zweck wird detailliert betrachtet.

Der neu entwickelte Algorithmus zur Lokalisierung von Solaranlagen klassifiziert jeden Pixel eines Orthophotos einzeln. Der Algorithmus basiert auf Faltungsoperationen, die die benachbarten Pixel mit einbeziehen, und einem neuronalen Netz. Um Ausreißer zu eliminieren werden verschiedene morphologische Operationen evaluiert.

Um den Rechenaufwand zu reduzieren, werden nur Bildbereiche betrachtet, die Gebäudedächern entsprechen. Dafür werden die Kartendaten herangezogen.

Mit dem Datensatz von den kalifornischen Orten konnte eine Genauigkeit von 72,2 % bei einer Trefferquote von 68,9 % erzielt werden. Der mit den gleichen Daten trainierte Algorithmus wurde auch mit einem eigens erstellten Datensatz zu einem deutschen Ort evaluiert. Dabei wurde eine Genauigkeit von 21,0 % bei einer Trefferquote von 45,7 % erreicht.

Die erzielten Ergebnisse und der eigens erstellte Datensatz wurde veröffentlicht, siehe [2].

Inhaltsverzeichnis

1 Einleitung

Der menschengemachte Klimawandel ist eines der wichtigsten Probleme, mit dem die Menschheit momentan zu kämpfen hat. Einen extrem wichtigen Betrag leistet dazu die Energiewende, welche den Umstieg von fossilen Brennstoffen auf Erneuerbare Energien umfasst. Diese Arbeit trägt dazu bei, den aktuellen Stand der Energiewende in Deutschland und potenziell auch in anderen Ländern im Hinblick auf PV-Anlagen einzuschätzen.

Über das Erneuerbare-Energien-Gesetz (EEG) regelt die Bundesrepublik Deutschland die Verwendung von erneuerbaren Energien. Mit diesem Gesetz soll unter anderem die Stromerzeugung durch Photovoltaik (PV) gefördert werden. Um den gesamten Energiebedarf weitestgehend aus erneuerbaren Energien zu decken, müsste eine Nennleistung zwischen 120 und 640 GW durch PV erzeugt werden (Zusammenfassung verschiedener Studien durch das *Fraunhofer-Institut für Solare Energiesysteme ISE*, vgl. [3]). „Ende 2019 waren in Deutschland PV-Module mit einer Nennleistung von knapp 49 GW installiert, verteilt auf über 1,7 Mio. Anlagen“, stellt das Fraunhofer ISE fest. Die Stromversorgung Deutschlands durch PV-Anlagen hat einen nennenswerten Anteil und steigt.

Wie Killinger in seiner Dissertation darlegt (vgl. [4]), ist es für die politischen Ziele der Versorgungssicherheit und Wirtschaftlichkeit (vgl. [5]) wichtig, die Energieerzeugung durch PV-Anlagen in Echtzeit und als Prognose präzise bestimmen zu können, weil die volatile Stromerzeugung durch PV die Stabilität des Stromnetzes gefährdet. Killinger stellt des Weiteren fest, dass den einzelnen Akteuren die kontinuierlichen Leistungsdaten nur von weniger als 25 % aller PV-Anlagen zur Verfügung stehen. Um die Leistungsdaten dennoch genau abschätzen zu können, werden verschiedenste Algorithmen und Datenquellen benötigt, darunter die exakten Positionen und Orientierungen aller beteiligten PV-Module.

Diese Arbeit beschäftigt sich damit, PV-Anlagen unter anderem für solche Zwecke zu lokalisieren. Aufgrund der großen Datenmengen muss das automatisiert passieren. Dafür werden verschiedene Datenquellen und Algorithmen evaluiert.

Das Marktstammdatenregister (MaStR) ist eine behördliche Datenbank, in der In-

Abbildung 1: Aufdachanlage: Photovoltaik-Module auf einem Dach (Bildquelle: [6])

formation über PV-Anlagen eingetragen werden müssen. Darin werden unter anderem die Koordinaten und Orientierung von gesamten PV-Anlagen eingetragen. Allerdings sind diese Informationen aus Datenschutzgründen bei den meisten PV-Anlagen nicht öffentlich.

Um die Daten dennoch für aggregierende Auswertungen wie von Killinger zur Verfügung zu stellen, beschäftigt sich diese Arbeit mit dem Lokalisieren von vorhandenen PV-Anlagen anhand von öffentlichen Datenquellen.

Wir beschränken uns dabei auf die Verarbeitung von digitalen Orthophotos (entzerrte, georeferenzierte Luftbilder), mit dem Ziel, auf Gebäudedächern installierte Solaranlagen zu lokalisieren. Dafür beziehen wir außerdem georeferenzierte Umringpolygone von Gebäudegrundrissen mit ein. Wir betrachten eine kleine Auswahl von vielen möglichen Algorithmen, mit denen man sich diesem Ziel nähern kann.

1.1 Photovoltaik und Solarthermie

Photovoltaik (PV) bezeichnet die Umwandlung von Lichtenergie in elektrische Energie mittels *Solarzellen*. Mehrere zusammengeschaltete Solarzellen bilden ein *Solarmodul*. Eine *Photovoltaik-Anlage* (PV-Anlage) besteht aus Solarmodulen. Eine *Aufdachanlage* ist eine PV-Anlage, die auf einem Hausdach installiert ist (siehe Abbildung 1). Eine andere verwandte Bauform ist die Indachanlage. Diese ersetzt im Gegensatz zu Aufdachanlagen die Dachverkleidung. Indachanlagen sind meist weniger effizient als Aufdachanlagen.

PV ist abzugrenzen von *Solarthermie*. Bei letzterer wird im Gegensatz zu PV die Sonneneinstrahlung zunächst in Wärme umgewandelt, welche dann zum Heizen oder zur Stromerzeugung verwendet werden kann. Ebenso wie PV-Anlagen können Solarthermie-Anlagen auch auf Gebäudedächern aufgebaut werden, in Form von *Solarkollektoren*. Letztere gibt es in verschiedenen Bauformen, darunter *Flachkollektoren* (siehe Abbildung 2).

Abbildung 2: Solarthermie: Flachkollektoren zur Warmwasserbereitung und Heizungsunterstützung (Bildquelle: [7])

Für eine genauere Abgrenzung siehe „Regenerative Energiesysteme“ [8].

In unserer Arbeit beziehen wir uns nur auf PV-Anlagen.

Dazu haben wir einen Datensatz an Orthophotos manuell in „enthält PV-Anlage“ und „enthält keine PV-Anlage“ eingeteilt. Wegen der verwendeten Bodenauflösung von bestens 10 cm pro Pixel und der großen optischen Ähnlichkeit von Solarmodulen und Solarkollektoren (vgl. Abbildung 1 und Abbildung 2) unterscheiden wir bei unseren manuell klassifizierten

Orthophotos nicht zwischen den beiden. Deswegen sind in unseren manuell erstellten Datensätzen möglicherweise Solarthermie-Anlagen fälschlicherweise als PV-Anlagen klassifiziert. Da wir in unserer Arbeit nicht darauf den Fokus legen wollen, nehmen wir an, dass diese Fehler nicht passiert seien. Deswegen ist im Nachfolgenden nur noch von PV-Anlagen die Rede.

1.2 Inhalt dieser Arbeit

Orthophotos und neuronale Netze bilden wichtige Grundlagen für die danach vorgestellten Themen, weswegen es dazu in Abschnitt 2 eine kurze Einführung gibt.

Mit der Lokalisierung von PV-Anlagen in Orthophotos haben sich bereits verschiedene wissenschaftliche Arbeiten beschäftigt, welche in Abschnitt 3 vorgestellt werden. Dort wird auch auf andere relevante Forschung eingegangen.

Danach wird der neu entwickelte Ansatz zur Lokalisierung von PV-Anlagen in Orthophotos vorgestellt. In Abschnitt 4 wird mit den möglichen und verwendeten Datenquellen dafür begonnen. Abschnitt 5 erläutert, wie die verwendeten Datenquellen für den in Abschnitt 6 beschriebenen Algorithmus vorbereitet werden.

Die damit erzielten Ergebnisse werden in Abschnitt 7 diskutiert. Abschnitt 8 fasst die gesamte Arbeit zusammen und gibt einen Ausblick, womit sich zukünftige Forschung beschäftigen könnte.

2 Grundlagen

Orthophotos als Datenquelle und neuronale Netze als Algorithmen sind essenzielle Grundlagen für unsere Arbeit und werden daher in den folgenden Abschnitten angesprochen.

2.1 Orthophotos: Gebäude aus der Vogelperspektive

Der Schwerpunkt unserer Arbeit liegt auf der Analyse von sogenannten Orthophotos. Was diese auszeichnet und welche Anforderungen wir an diese stellen, wird im Folgenden beschrieben. Woher sie bezogen werden können, wird in Abschnitt 4.2 untersucht.

2.1.1 Eigenschaften und Anforderungen

Orthophotos sind Landschaftsfotos aus der Vogelperspektive (für gewöhnlich mit Flugzeugen aufgenommen). Sie zeichnen sich dadurch aus, maßstabsgetreu und von perspektivischen Verzerrungen befreit zu sein (siehe Abbildung 3). Digitale Orthophotos (kurz: DOP) werden aus Fotos und dreidimensionalen Höhenmodellen erstellt. *TrueDOP* werden Bilder genannt, die mit einer besseren Technik erstellt werden, durch die Umklappeffekte (Verkippungen) eliminiert werden. Dadurch lässt sich aus dem Bild keine Höheninformation mehr entnehmen und Objekte können fransige Ränder bekommen. Dafür verdeckt kein Objekt ein anderes und jedem Pixel des *TrueDOP* können Koordinaten korrekt zugeordnet werden. Diese Zuordnung, genannt *Georeferenzierung*, ist bei allen Orthophotos gegeben. (vgl. [9]–[12])

Die Qualität von Orthophotos hängt unter anderem von der sogenannten *Bodenauflösung* ab. Die Bodenauflösung (auch Ground Sample Distance, GSD genannt) beschreibt die Kantenlänge eines Quadrats Boden, welches auf genau einen Pixel abgebildet wird (vgl. [11]). Aufgrund der Erfassungstechniken kann der Fehlerbereich bei der Georeferenzierung größer sein als die Bodenauflösung (vgl. [14]–[16]).

Bei der Verwendung von Orthophotos sind meteorologische Effekte zu beachten. Da je nach Jahreszeit das Laub an den Bäumen und der Stand der Sonne variiert, hat dies Einfluss auf die in den Luftbildern dargestellten Schatten und Farbtemperaturen. Bei

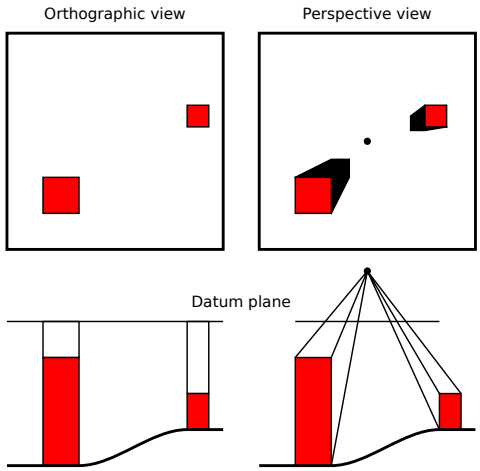

Abbildung 3: Orthophotos sind Bilder aus der Vogelperspektive, bei denen die perspektivische Verzerrung herausgerechnet wurde. (Bildquelle: [13])

Orthophotos, die zu verschiedenen Zeiten (und von verschiedenen Gegenden) erstellt wurden, kann dies die Vergleichbarkeit beeinträchtigen.

2.2 Neuronale Netze

Ein neuronales Netz ist eine populäre Form des maschinellen Lernens (vgl. [17]). Es verarbeitet eine Menge von Eingabewerten zu einer Menge von Ausgabewerten, was z.B. ein Foto und eine Klassifikation in Form einer Wahrscheinlichkeit pro Pixel sein kann.

Wie ein neuronales Netz eine Ausgabe berechnet, hängt in der Regel von sehr vielen Parametern ab. Um für diese Parameter möglichst geeignete Werte zu finden, wird das Netz „trainiert“. Das geschieht im Fall von vollüberwachtem Lernen („supervised“)[1], in dem die Ausgaben von verschiedenen Eingaben („Trainings-Daten“) berechnet und bewertet werden. Dafür muss zu den Trainingsdaten der Sollzustand der Ausgabe bekannt sein. Die Abweichung von der gewünschten Ausgabe kann über einen einzelnen Wert, berechnet über eine sogenannte *Loss*-Funktion, bestimmt werden. Anhand dessen

[1] Neben vollüberwachtem Lernen gibt es z.B. auch teilüberwachtes und unüberwachtes Lernen. Bei ersterem werden mit wenigen Eingabedaten neue Eingabedaten berechnet. Bei letzterem wird kein Sollzustand der Ausgabe vorgegeben.

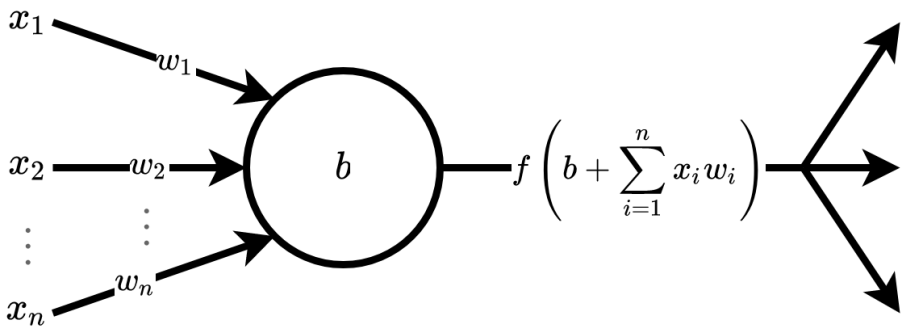

Abbildung 4: Ein Neuron.

werden die Parameter angepasst, sodass die Ausgabe eher dem gewünschten Zustand entspricht (der *Loss* wird optimiert). Um Rechenaufwand zu sparen, muss die Optimierung nicht nach jeder Eingabe gemacht werden. Es ist üblich, mehrere Trainingsdaten („Batch“) einzugeben und die Optimierung dann anhand der gemittelten Ausgabe anzuwenden. Das einmalige Trainieren mit allen Daten eines Datensatzes wird Epoche genannt.

Ein *Neuron* ist der elementare Baustein zum Entwerfen von neuronalen Netzen. Es ist eine Funktion, die mindestens eine Eingabe zu einer Ausgabe verrechnet, siehe Abbildung 4. Jede Eingabe x_i wird mit einem Faktor w_i gewichtet. Die Summe dieser Produkte, zuzüglich einem Schwellwert b („bias“) wird in eine *Aktivierungsfunktion* f gegeben. Diese ist monoton steigend, ansonsten aber frei wählbar. Üblich sind Aktivierungsfunktionen, die die Eingabe auf einen festen Wertebereich abbilden. Der Wert der Aktivierungsfunktion ist die Ausgabe, genannt *Aktivierung*, des Neurons.

Ein neuronales Netz enthält für gewöhnlich viele Neuronen. Diese werden in geordnete *Layer* gruppiert, vgl. Abbildung 5. Die Aktivierungen von den Neuronen eines Layers werden als Eingabe der Neuronen des nächsten Layers verwendet.[2] Der erste Layer bildet die Eingabe und der letzte die Ausgabe des neuronalen Netzes. Alle Layer dazwischen werden aufgrund dieser Eigenschaft als *Hidden Layer* bezeichnet. Die Anzahl Neuronen pro Layer kann innerhalb eines Netzes variieren.

[2]Es gibt auch davon abweichende Architekturen, z.B. rekurrente neuronale Netze.

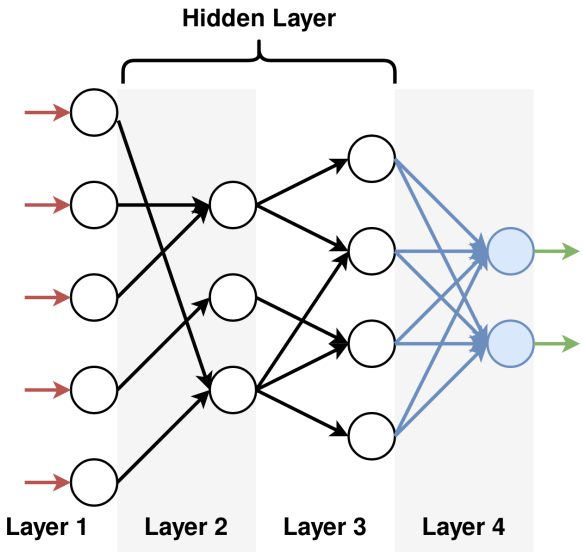

Abbildung 5: Beispiel für ein neuronales Netz mit 4 Layern. Der erste Layer hat 5 Neuronen, der zweite 3, der dritte 4 und der letzte 2. Der letzte Layer (blau) ist vollverbunden. Das Netz hat 5 Eingabewerte (rot) und 2 Ausgabewerte (grün).

Weitere Grundlagen zu neuronalen Netzen können beispielsweise in [18] gefunden werden.

2.2.1 Convolutional Neural Networks

Im folgenden wird eine von der allgemeinen Form abweichende Architektur namens *Convolutional Neural Network (CNN)* vorgestellt. Diese ist in der Bildverarbeitung beliebt (vgl. z.B. [19]–[21]).

Gegenüber den oben beschriebenen neuronalen Netzen kommen in den CNN unter anderem Faltungsoperationen („convolution“) hinzu. Bei einer Faltung wird jeder Datenpunkt unter Einbezug seiner benachbarten Datenpunkte zu einem neuen Datenpunkt umgerechnet. Abbildung 6 veranschaulicht eine Faltung einer Rastergrafik. Eine Faltungsmatrix (blau) wird auf einen Bereich um einen Pixel (gelb/rot) angewendet. Das Ergebnis der Operation ist ein einzelner Wert, der an der der Eingabegrafik entsprechenden Stelle in der Ausgabegrafik übernommen wird (grün/violett). Für jeden so errechneten Wert muss die Umgebung in der Eingabegrafik existieren oder extrapolier-

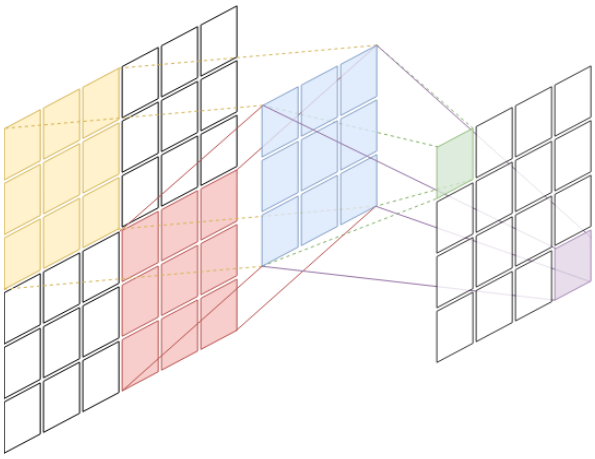

Abbildung 6: Schematisches Beispiel für eine Faltungsoperation auf einer Rastergrafik, wie sie bei CNNs verwendet wird.

bar sein. Die Faltung muss nicht für jeden Pixel der Eingabegrafik angewendet werden. Die Größen von Ein- und Ausgabegrafik können sich unterscheiden.

Beim Trainieren passt das CNN die Werte der Faltungsmatrix an.

Ein weiteres nützliches Werkzeug von CNNs sind sogenannte *Pooling Layer*. Diese reduzieren die Anzahl Werte (Neuronen) für den nachfolgenden Layer, wodurch irrelevante Details gefiltert werden sollen. Möglich sind zum Beispiel *Max-Pooling-Layer*, die eine Umgebung von Werten durch ihren maximalen Wert ersetzen (vgl. [22]).

3 Stand der Forschung

In den folgenden Abschnitten betrachten wir die für unsere Arbeit relevante Forschung. Das betrifft besonders Arbeiten, die versuchen, mithilfe von maschinellem Lernen Solaranlagen in RGB-Orthophotos zu erkennen, siehe dazu Abschnitte 3.2 bis 3.6. Keine dieser Arbeiten versucht, Solaranlagen *in Deutschland* zu lokalisieren – dazu konnten wir keine Literatur finden. Die erforschten Methoden lassen sich aber wahrscheinlich auch auf Datensätzen von deutschen Regionen anwenden.

Es wurde jedoch sehr wohl bereits daran geforscht, Solaranlagen in Deutschland zu lokalisieren, beispielsweise anhand von Infrarot-Luftbildaufnahmen: Dotenco et al. haben das 2016 erfolgreich versucht (vgl. [23]). Die Infrarotbilder haben sie eigens 2015 mit einer tief fliegenden Drohne erstellt. Da wir uns vorab auf RGB-Orthophotos als Datenmaterial festgelegt haben, gehen wir auf die Analyse von Infrarotbildern nicht weiter ein.

Als Beispiel für den Nutzen unserer Arbeit wird in Abschnitt 3.1 die Dissertation von Killinger (vgl. [4]) aufgegriffen.

Unsere Arbeit beschäftigt sich damit, bestehende Solaranlagen zu lokalisieren. Davon abzugrenzen sind viele Arbeiten, die sich mit dem Potential für Solar- oder speziell PV-Anlagen beschäftigen. Damit hat sich beispielsweise Fath in ihrer Dissertation beschäftigt und eine ausführliche Literaturrecherche angestellt (vgl. [24]).

3.1 Dissertation von Killinger (2018)

In seiner Dissertation (vgl. [4]) beschäftigt sich Killinger mit Methoden zur Hochrechnung der Energieerzeugung durch PV-Anlagen auf Basis von Referenzmessungen und Geodaten. Die Hochrechnungen sollen in Echtzeit und als Prognose möglich sein.

Für dieses Ziel benötigt Killinger die Standorte von PV-Modulen. Um seine Hochrechnungen zu verbessern, lässt Killinger die Modulorientierung, also die Neigung und Drehung der PV-Module, in die Berechnungen mit einfließen. Der Mehrwert ist signifikant, aber auch abhängig von den meteorologischen Bedingungen.

Zum Zeitpunkt der Dissertation wurden die Standorte, aber nicht die Modulausrich-

tungen von PV-Anlagen im EEG-Anlagenregister erfasst.[3] Killinger merkt außerdem an, dass der Informationsgehalt aus Datenschutzgründen reduziert wurde und bemängelt eklatante Qualitätsprobleme in den Stammdaten des EEG-Anlagenregisters. Mittlerweile wurde das EEG-Anlagenregister durch das Marktstammdatenregister ersetzt.[4] In diesem wird auch die Modulorientierung erfasst. Aus Datenschutzgründen sind jedoch die Standorte von lediglich einer Promille der PV-Anlagen öffentlich einsehbar (vgl. Abschnitt 4.1). Ohne die Korrelation mit dem Standort der PV-Anlage wird die Modulorientierung unbrauchbar. Daher werden alternative Wege zur Bestimmung der Modulorientierung benötigt.

3.1.1 Modulorientierung

Killinger hat einen Algorithmus zur Schätzung der Modulausrichtung entwickelt, welcher im Folgenden vorgestellt wird.

Die grundlegende Annahme ist, dass die Modulorientierung der Orientierung des zugrundeliegenden Daches entspricht.[5] Deshalb ist das neue Ziel, die entsprechende Dachorientierung zu bestimmen.

Der Standort einer PV-Anlage ist in Form der Anschrift des Eigentümers gegeben, bezogen aus Stammdaten von dem Stromnetzbetreiber *TransnetBW*[6]. Die PV-Anlage befindet sich nicht immer auf dem nächstgelegenen Dach zu dieser Anschrift, aber hinreichend oft, sodass Killinger diesen Ansatz verfolgt.

Die Anschrift wird mithilfe einer API[7] von *Google Maps* in Geo-Koordinaten umgewandelt. Zu diesen wird dann der nächstgelegene Gebäudeumriss im Datensatz von *OpenStreetMap* (vgl. [25]) gesucht.

Anhand von digitalen Höhenmodellen (DEM) von der Stadt *Freiburg im Breisgau* (vgl. [26]) werden die Dachflächen samt Orientierung innerhalb des Gebäudeumrisses

[3] vgl. *Anlagenregisterverordnung* (außer Kraft)

[4] vgl. *Verordnung über die Registrierung energiewirtschaftlicher Daten* vom 10. April 2017

[5] Diese Annahme funktioniert nicht für Flachdächer. Auf diesen sind PV-Anlagen üblicherweise zwecks Effizienz abweichend vom Dach orientiert.

[6] Killinger hat sich bei der Schätzung von Modulorientierungen auf den Stadtkreis Freiburg beschränkt, weswegen der nicht bundesweit agierende Stromnetzbetreiber als Datenquelle genutzt werden konnte.

[7] *Application Programming Interface* (Programmierschnittstelle)

bestimmt. Diese Dachflächen werden gefiltert auf jene, die groß genug für die installierte PV-Leistung sind (Annahme: $10\frac{\mathrm{m}^2}{\mathrm{kWp}}$). Davon wird dann die Dachfläche mit der für PV-Anlagen besten Orientierung gewählt.

Ergebnisse Bei der Überprüfung konnte Killinger feststellen, dass Wohnhäuser mit Satteldächern unterrepräsentiert waren. Stattdessen kamen viele komplexere Dächer vor. Er empfiehlt deswegen und wegen des hohen Einflusses auf die beabsichtigten Hochrechnungen, die Modulausrichtung von großen Freiflächen- und Flachdachanlagen manuell zu bestimmen.

Die Position der gefundenen Dachflächen zeigte eine hohe Übereinstimmung mit entsprechenden Luftbildern. Für die Validierung werden die tatsächlichen Modulorientierungen benötigt.

Für Killingers Methode werden aktuelle DEM vorausgesetzt, deren (nach seiner Aussage) Verfügbarkeit, Aktualität und Bepreisung nach Bundesland variieren. Liegen keine aktuellen DEM vor, empfiehlt er eine alternative Methode von Mainzer et al. und Fichtner et al. (vgl. [27], [28]). Bei dieser werden die Modulorientierungen anhand von Orthophotos und Gebäudeumrissen aus *OpenStreetMap* berechnet.[8]

Killinger erwähnt, dass mit den DEM außerdem die Berechnung von Verschattungen der PV-Module möglich gewesen wäre, er aber zur Einsparung von Rechenzeit davon abgesehen habe.

3.1.2 Erkenntnisse für diese Arbeit

Killingers Dissertation zeigt den Bedarf, PV-Anlagen zu lokalisieren. Neben den reinen Standorten sind außerdem die Modulorientierungen wertvoll.

In dieser Arbeit versuchen wir, die PV-Module pixelgenau (mit 10 cm Bodenauflösung) zu lokalisieren, was exakter ist, als die von Killinger verwendeten Anschriften. Das birgt das Potential, nicht nur die Standorte, sondern mithilfe derer auch die Modulorientierungen präziser zu bestimmen.

[8] „Aktuell ist der Ansatz leider noch auf bestimmte Dachtypen begrenzt und der Neigungswinkel wird auf Basis einer Normalverteilung geschätzt.“ (vgl. [4] S. 158)

3.2 Forschung von Malof et al. (2015, 2016)

In mehreren Veröffentlichungen beschäftigen sich Malof et al.[9] mit der automatisierten Lokalisierung von PV-Modulen in Orthophotos. Sie haben 2015 und 2016 Arbeiten mit ähnlichen Titeln veröffentlicht, welche die gleiche Problemstellung auf verschiedene Arten lösen. (vgl. [29]–[32]) Diese werden im Folgenden umrissen.

3.2.1 Support Vector Machines (2015)

In ihrer 2015 veröffentlichten Arbeit (vgl. [30]) haben Malof et al. eine Software entwickelt, die auf Hausdächern installierte Solaranlagen zählt. Über ein reines Zählen hinausgehende Analysen waren außerhalb der Zielsetzung – der allgemeine Ansatz sollte auf Machbarkeit geprüft werden.

Datenquelle Die analysierten Bilder waren 2014 aufgenommene RGB-Orthophotos von 100 kalifornischen Hausdächern mit einer Bodenauflösung von 30 cm. Auf 50 % der Bilder waren Hausdächer mit Solaranlagen abgebildet, auf den anderen solche ohne.

Algorithmus Der Algorithmus verwandelt ein Bild in eine Menge von Regionen (benachbarte Pixel), zugeordnet mit Wahrscheinlichkeiten, zu denen es sich bei den Regionen um Solaranlagen handelt. Der Algorithmus konvertiert das Bild zunächst in Graustufen, bestimmt darin *Maximally Stable Extremal Regions* (vgl. [33]) und filtert diese anhand von manuell unter Betrachtung des Datensatzes ermittelten Kriterien. Zu diesen Regionen wurden je verschiedene *Features* berechnet, basierend auf Farben, Graustufen und Form der Region. Diese Features werden von einer *Support Vector Machine* (vgl. [34]) auf eine Wahrscheinlichkeit pro Region, ob es sich um eine Solaranlage handelt, heruntergebrochen. Zum Schluss wurden noch eng benachbarte Regionen mit einem *Mean-Shift-Clustering*-Algorithmus (vgl. [35]) mit Gauss-Filter zusammengefasst.

In dem Datensatz wurden manuell die Konturen der Solaranlagen markiert. Eine von der Software markierte Region wurde als korrekt gewertet, wenn mindestens 50%

[9] Malof, Bradbury, Collins und Newell sind bei allen Arbeiten als Autoren genannt. Außer diesen werden je nach Veröffentlichung noch verschiedene weitere Autoren genannt.

davon auf der Solaranlage lag und mindestens 10% der Solaranlage abgedeckt wurde. Die Ausgabequalität der Support Vector Machine wurde mit Leave-One-Out Kreuzvalidierung[10] bestimmt.

Ergebnisse Mit diesem Algorithmus haben Malof et al. in einem Datensatz aus vorausgewählten Hausdach-Bildern 94 % von 53 Solaranlagen erkannt (bei 4 False Positives). Vor Anwendung der Support Vector Machine wurde diese Erkennungsrate bei einer Fehlerquote von 175 False Positives pro km^2 erzielt. Nach Anwendung des vollständigen Algorithmus lag die Fehlerquote bei 20 False Positives pro km^2.

3.2.2 Random Forest Classifier (2016)

2016 haben Malof et al. (nicht die gleichen *et al*!) eine weitere Arbeit veröffentlicht (vgl. [29]), in der ein anderer Datensatz und ein anderer Algorithmus zum Erkennen von Solaranlagen in Orthophotos eingesetzt wurde.

Datenquelle Sie haben dafür 2013 erstellte Orthophotos mit 30 cm Bodenauflösung von einer 135 km^2 großen Fläche in Fresno, Kalifornien, USA genommen. In diesen Bildern haben sie 2794 PV-Anlagen händisch markiert. Diesen Datensatz haben Malof et al. zusammen mit weiteren markierten Satellitenbildern von anderen Orten veröffentlicht (vgl. [1]).

Ihren Datensatz haben Malof et al. im Verhältnis 2:1 in Trainings- und Test-Daten aufgeteilt. Erstere haben sie zum Optimieren ihres Algorithmus verwendet, letztere zum Testen, wie gut der Algorithmus funktioniert.

Algorithmus Die Solarmodul-Erkennung funktioniert in vier Schritten:

1. Das Bild wird von den 3 RGB-Kanälen pro Pixel auf 102 „Features“ pro Pixel umgerechnet, indem pro Pixel und Farbkanal das arithmetische Mittel und die

[10] Bei einer *Leave-One-Out Kreuzvalidierung* wird ein Algorithmus mit einem Element des Datensatzes (hier einzelne Hausdach-Bilder) getestet, nachdem er mit allen übrigen Elementen trainiert wurde. Das wird für jedes Element im Datensatz wiederholt und die Ergebnisse werden gemittelt.

Varianz von umliegenden 3x3-Pixelgruppen berechnet werden. Die genaue Zusammensetzung wird in Abschnitt 6.1 erläutert.

2. Ein *Random Forest Classifier* (vgl. [36]) macht aus den 102 Features pro Pixel eine Wahrscheinlichkeit pro Pixel, die angibt, ob dieser Pixel zu einem PV-Modul gehört.

3. Ein *Post-Processing*-Algorithmus glättet die Ausgabedaten des Random Forest Classifiers, sucht Regionen von benachbarten Pixeln mit hoher Wahrscheinlichkeit und setzt die Wahrscheinlichkeit von Pixeln außerhalb dieser Regionen auf Null. Zur Bildung von Regionen werden *Otsus* Algorithmus (vgl. [37]) und die morphologischen Operationen *Closing* und *Opening* (vgl. Abschnitt 6.3) in dieser Reihenfolge angewendet.

4. Im letzten Schritt werden Regionen von Pixeln mit hoher Wahrscheinlichkeit als solche zusammengefasst und mit der größten darin vorkommenden Wahrscheinlichkeit beschriftet.

Ergebnisse Malof et al. haben ihren Algorithmus mithilfe von Genauigkeit-Trefferquote-Diagrammen bewertet. Darin wird der Anteil der korrekt klassifizierten an allen tatsächlichen Solaranlagen gegenüber dem Anteil der korrekt klassifizierten an allen (auch falsch) klassifizierten Solaranlagen dargestellt. Diese Diagramme haben sie für die Ergebnisse des Algorithmus nach dem Random Forest Classifier (Pixel-basierte Ergebnisse) und nach dem Post-Processing (Region-basierte Ergebnisse) erstellt, siehe Abbildung 7. Um bei Letzterem zu bewerten, ob ein eine vom Algorithmus erkannte Solaranlage einer von einem Menschen erkannten Solaranlage entspricht, wurde der Jaccard-Index[11] verwendet: Sofern sich die von Mensch und Algorithmus ausgezeichneten Bereiche zu einem Mindestanteil überlappen, wurde das Ergebnis des Algorithmus als korrekt gewertet. Im Diagramm sind verschiedene Kurven für verschiedene Mindestanteile dargestellt.

[11] Der Jaccard-Index wird durch $\frac{\text{True Positive}}{\text{True Positive+False Positive+False Negative}}$ berechnet.

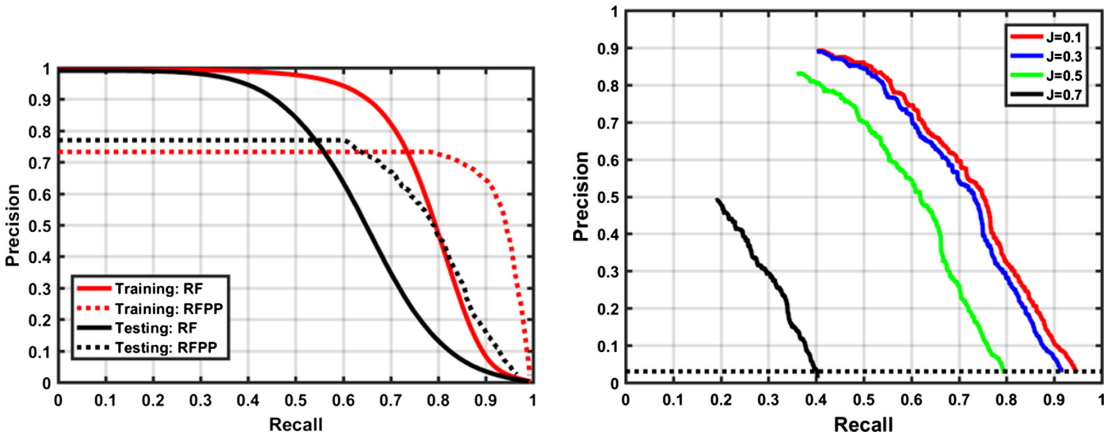

Abbildung 7: Dies sind die Genauigkeiten über Trefferquoten, die Malof et al. mit Random Forest Classifieren erreichen konnten. Links sind die Pixel-basierten Ergebnisse auf Trainings- und Testdaten vor (durchgezogenen Linien) und nach (gestrichelte Linien) Post-Processing zu sehen. Rechts sind die Region-basierten Ergebnisse für verschiedene Schwellwerte des Jaccard-Indexes J abgebildet. (vgl. Abschnitt 3.2.2, Bildquelle: [29])

Bei allen Bewertungsmethoden hat der Algorithmus deutlich besser als ein rein zufälliger Algorithmus abgeschnitten. Der Random Forest Classifier hat auf den Trainingsdaten bessere Ergebnisse als auf den Testdaten erzielt, was ein Zeichen von Überanpassung („Overfitting“) ist.

Insgesamt war der Algorithmus gut darin, einzelne Pixel als Bestandteil einer Solaranlage und ganze Solaranlagen zu erkennen. Allerdings war der Algorithmus deutlich schlechter darin, die Form und Größe von Solaranlagen zu bestimmen. Je präziser die gefundene Form und Größe einer Solaranlage mit der Realität übereinstimmen sollte (höherer Jaccard-Index), desto schlechter war die Ausgabequalität des Algorithmus. Malof et al. schlagen vor, mit ihrem Algorithmus den nach Solaranlagen zu durchsuchenden Bildbereich für aufwändigere und präzisere Algorithmen einzugrenzen.

Sie beschreiben ihre Arbeit als Machbarkeitsstudie – ob die erzielte Qualität der Ausgabedaten nützlich ist, und wie weitere Informationen aus den Bilddaten extrahiert werden können, bleibt eine offene Frage. Dass ihre Forschung generell Nutzen trägt, sind sich Malof et al. sicher. Um zukünftige Forschung auf diesem Gebiet zu fördern, haben Malof et al. ihren Datensatz veröffentlicht (siehe [1]).

3.2.3 Evaluation von Feature-Vektoren (2016)

In einer weiteren Arbeit (vgl. [32]) gehen Malof et al. genauer auf die in der vorherigen Arbeit verwendeten Feature-Vektoren für den *Random Forest Classifier* ein. Sie vergleichen vier verschiedene Berechnungen (und Kombinationen) von Feature-Vektoren, die im Folgenden beschrieben werden. Ihre Ergebnisse sind in Abbildung 8 dargestellt.

Als weiteren Forschungsbedarf nennen Malof et al. eine Verringerung des Rechenaufwands und den Einbezug eines größeren Kontexts zu den Pixeln.

Rohe Pixel Als Referenzmessung für die folgenden Feature-Vektoren haben Malof et al. einen Feature-Vektor erstellt, indem sie aus einer 7×7-Pixel-Umgebung die RGB-Werte unverändert übernommen haben. Die Größe der Umgebung haben sie als Mittelweg zwischen Informationsgehalt und Rechenaufwand gewählt.

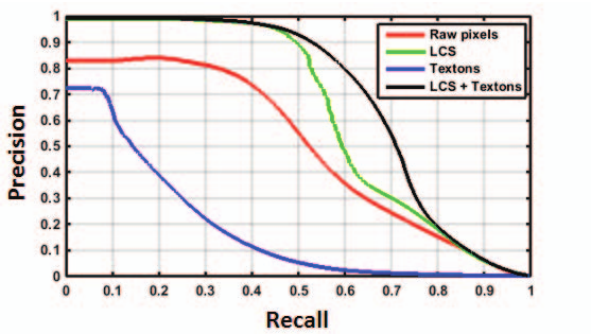

Abbildung 8: Ergebnisse, die Malof et al. mit einem *Random Forest Classifier* und verschiedenen Feature-Vektoren erzielt haben. *Precision* ist die Genauigkeit und *Recall* die Trefferquote. (vgl. Abschnitt 3.2.3, Bildquelle: [32])

Lokale Farbstatistiken (LCS) Dieser Feature-Vektor entspricht genau dem in der vorherigen Arbeit [29] verwendeten Feature-Vektor: Von verschiedenen Pixeln in der Umgebung werden arithmetisches Mittel und Varianz gebildet. Dieser Feature-Vektor wird im Rahmen der von uns verwendeten Feature-Vektoren in Abschnitt 6.1 detailliert beschrieben und in Abbildung 17 veranschaulicht.

Das hat bessere Ergebnisse als die rohen Pixel erzielt.

Textons Dieser Feature-Vektor ist ein Versuch, Formen und Texturen zu erkennen. Deshalb wird das Eingabebild vorab zu Graustufen konvertiert. Bestimmte Formen und

Texturen werden über sogenannte *Textons* repräsentiert. Jeder Pixel in einer 9×9-Umgebung wird einem Texton zugeordnet. Der Feature-Vektor ist dann ein Histogramm, das für jedes Texton angibt, wie viele Pixel diesem zugeordnet wurden.

Dafür werden Algorithmen benötigt, um alle möglichen Textonen zu bestimmen, eine praktikable Teilmenge davon auszuwählen und einzelne Pixel Textonen zuzuordnen.

Malof et al. wenden auf das Graustufenbild mehrere *Leung-Malik*-Filter (vgl. [38]) mit verschiedenen Parametern an, wodurch sie 36 Werte pro Pixel erhalten, ein „Texton“. Um die Textonen für das Histogramm zu auszuwählen, wenden sie diese Filter auf eine Million Trainings-Pixel an, 50 % davon entsprechen einem PV-Modul. Mit einem *k-Means*-Algorithmus wählen sie davon 30 repräsentative Textons aus, welche für das Histogramm verwendet werden.

Diese Feature-Vektoren benötigen somit eine eigene Trainings-Phase.

Dieser Feature-Vektor hat im Vergleich die schlechtesten Ergebnisse geliefert. Malof et al. deuten daraus, dass die hierbei vernachlässigten Farbinformationen ausschlaggebend sind.

Input (40 x 40 RGB image)
conv3-32
conv3-32
maxpool
conv3-32
conv3-32
maxpool
conv3-64
conv3-64
maxpool
fully connected - 64
Output (2-way soft-max)

Abbildung 9: Dies ist die Architektur des von Malof et al. verwendeten CNNs. `conv3-32` steht für 32 Faltungsoperationen auf einer 3×3-Umgebung. `maxpool` steht für einen *Max-Pooling-Layer* (vgl. [22]). Der letzte Layer besteht aus 64 vollverbundenen Neuronen. (Bildquelle: [31])

Lokale Farbstatistiken und Textons (LCS+Textons) Malof et al. haben die Feature-Vektoren *LCS* und *Textons* zu einem gemeinsamen Feature-Vektor zusammengefügt. Damit haben sie im Vergleich die besten Ergebnisse erzielt.

3.2.4 Convolutional Neural Network (2016)

Neben den bisher zusammengefassten Ansätzen haben Malof et al. auch ein *Convolutional Neural Network* (CNN, in Abschnitt 2.2.1 erläutert) zur Lokalisierung von PV-Anlagen in Orthophotos ausprobiert (vgl. [31]). Naheliegenderweise haben sie ihre Datenquelle wiederverwendet (vgl. Abschnitt 3.2.2).

Weil CNNs für gewöhnlich hohen Rechenaufwand beim Trainieren haben, verwenden Malof et al. das CNN nur zu Verbesserung der Genauigkeit der Ergebnisse. Der in der vorangegangenen Arbeit entworfene *Random Forest Classifier* (RF, vgl. Abschnitt 3.2.2) mit *LCS*-Feature-Vektor (vgl. Abschnitt 3.2.3) klassifiziert zunächst die Eingabedaten. Die als entspricht-einem-PV-Modul (positiv) klassifizierten Pixel bekommt dann das CNN zu sehen, um daraus falsche positive Klassifizierungen zu filtern.

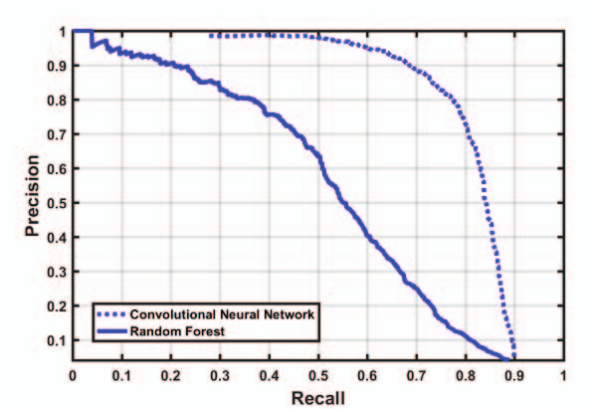

Abbildung 10: Mit einem CNN konnten Malof et al. die Genauigkeit eines Random Forest Classifiers verbessern. (Bildquelle: [31])

Das CNN klassifiziert im Gegensatz zum RF keine einzelnen Pixel, sondern 40×40-Pixel-Umgebungen. Ausgabe dazu ist je eine Wahrscheinlichkeit, ob ein PV-Modul zu sehen ist, oder nicht (*2-Wege Softmax*). Zum Lernen des CNNs verwenden Malof et al. einen *Stochastic Gradient Descent*. Sie rotieren die positiven Trainingsdaten in 45°-Schritten, zwecks mehr Variation. Als Aktivierungsfunktion verwenden sie ReLU. Die Architektur des CNN ist in Abbildung 9 dargestellt.

Mit dieser Methodik konnten Malof et al. die Genauigkeit im Vergleich zum RF allein deutlich verbessern, siehe Abbildung 10.

Der RF hat oft mehrere PV-Module erkannt, wo aber nur *eins* war. Davon wurde nur die maximale Erkennungswahrscheinlichkeit in die Ergebnisse aufgenommen. Das CNN hat sich nach 30 Epochen nicht mehr verbessert, woraus Malof et al. schließen, dass

sie in ihren Testdaten PV-Module hatten, die sich so von denen in den Trainingsdaten unterscheiden, dass ein Lernen davon nicht möglich war. Überanpassung (Overfitting) scheine nicht aufgetreten zu sein.

3.3 Forschung von Puttemans et al. (2016)

Die Belgischen Forscher Puttemans et al. haben sich in ihrer 2016 veröffentlichten Arbeit ebenfalls mit der Lokalisierung von PV-Anlagen anhand von Orthophotos beschäftigt (vgl. [39]). Als Motivation geben sie an, dass ein Stromnetzbetreiber durch das Aufspüren von nicht registrierten PV-Anlagen finanzielle Betrüge entlarven möchte.

3.3.1 Datenquelle

Puttemans et al. verwenden RGB-Orthophotos von der *Flanders Geographical Information Agency* mit einer Bodenauflösung von 25 cm. Wann diese Bilder aufgenommen wurden, geben Puttemans et al. nicht an. In der Flämischen Region haben sie 2500 Solaranlagen als solche klassifiziert. Außerdem haben sie für eine $4\,\mathrm{km}^2$-große Fläche der Innenstadt von *Sint-Truiden* alle 313 Solaranlagen pixelgenau annotiert. Vor dieser Annotation haben sie die Bilder mit einem bi-kubischen Operator auf die doppelte Seitenlänge vergrößert, um mehr Pixel pro Solarmodul zu erhalten. Diesen Datensatz haben sie veröffentlicht (vgl. [40]).

Mittlerweile seien Orthophotos mit 8 cm Bodenauflösung verfügbar, was die Genauigkeit der Algorithmen verbessern könnte.

Als allgemein den Orthophotos und der Zielsetzung inhärente Schwierigkeiten nennen Puttemans et al.:

- Lichtreflexionen, in Abhängigkeit des Sonnenscheins, verändern das Aussehen von Solarmodulen stark.
- Die Orientierung (Himmelsrichtung) und Neigung (Dachschräge) von Solarmodulen variiert und verdreht bzw. verzerrt deren Abbildung in Orthophotos.
- Je nach Material sehen Solarmodule verschieden aus.

3.3.2 Algorithmus

Puttemans et al. haben mehrere Algorithmen evaluiert, welche im Folgenden vorgestellt werden. Alle Algorithmen geben pro Pixel eine Wahrscheinlichkeit aus, ob es sich dabei um ein Solarmodul handelt. Weil die Eingabebilder skaliert wurden, werden die Ausgabedaten dann wieder herunter gerechnet, auf die Hälfte der ursprünglichen Seitenlänge. Die Wahrscheinlichkeiten werden anschließend über einen anpassbaren Schwellwert zu binären Aussagen gemacht. Die Ergebnisse verbessern Puttemans et al. mit den morphologischen Operationen *Opening* und *Closing* (vgl. Abschnitt 6.3), Kontur-Erkennung und -füllung.

Support Vector Machine (HSV) Sie transformieren die RGB-Bilder in den HSV-Farbraum[12], weil sich darin Farben besser unterscheiden lassen. Aus der Datenquelle haben sie positive Pixel (einem Solarmodul entsprechend) und negative Pixel (keinem Solarmodul entsprechend) im Verhältnis 1:2 ausgewählt. Bei der Auswahl der Positiv-Beispiele haben Puttemans et al. die andersfarbigen Ränder von PV-Modulen außen vor gelassen, um ein kleineres Farbspektrum zu bekommen.

Mit diesen Pixeln trainieren sie eine *Support Vector Machine* mit linearem Kernel.

Maximally Stable Extremal Regions (MSER) Mit *Maximally Stable Extremal Regions* (MSER, vgl. [33]) wird in Graustufenbildern nach Regionen gesucht, die über verschiedene Schwellwerte hinweg möglichst konsistent auftreten. Die Regionen werden dann je über eine Ellipse approximiert.

Anstatt die Eingabedaten zu Graustufen zu konvertieren, nehmen Puttemans et al. den blauen Farbkanal als Eingabe für den *MSER*–Algorithmus, weil Solaranlagen einen hohen Blauanteil haben. Die Ausgabe davon filtern sie auf Ellipsen mit zu Solarmodulen passenden Größen und Achsen-Verhältnissen.

Danach filtern sie die Ausgabe mit dem zuvor vorgestellten *HSV+SVM*-Algorithmus.

Boosted Cascade (V&J) Aufbauend auf einem bestehenden Framework (vgl. [41]) verwendet dieser Algorithmus eine *boosted* (vgl. [42]) Kaskade von schwachen Klassifi-

[12]Der HSV-Farbraum hat Kanäle für den Farbwert, die Farbsättigung und die Helligkeit.

katoren (einfache Entscheidungsbäume). Der Algorithmus verwertet die Struktur und Textur der Bilder anhand von *local binary pattern features* (vgl. [43]). Der Algorithmus operiert auf Graustufenbildern, wofür Puttemans et al. wie bereits beim *MSER*-Algorithmus den blauen Farbkanal nehmen.

Der Algorithmus ist abhängig von der Orientierung der Solarmodule, weswegen Puttemans et al. die Orientierung der Trainingsdaten harmonisiert und die der Testdaten variiert haben. Die Verdrehung der Testdaten wird nach der Ausgabe wieder umgekehrt.

Aggregate Channel Features (ACF) Auf bei diesem Algorithmus setzen Puttemans et al. auf ein bestehendes Framework (vgl. [44]). Dieses berechnet diverse Features und bestimmt automatisch, welche davon am geeignetsten für die gegebene Aufgabe sind. Welche das in ihrem konkreten Fall sind, geben Puttemans et al. nicht an. Der selbst implementierte Algorithmus war mangels Parallelisierung nicht sehr zeiteffizient.

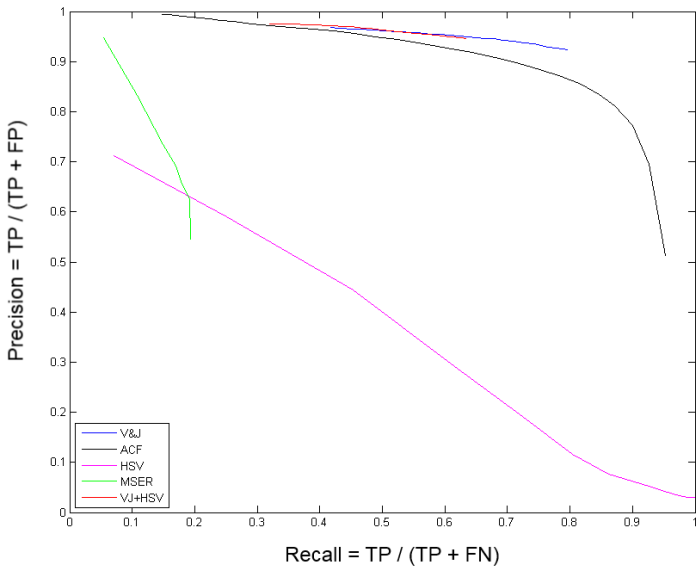

Abbildung 11: Hier wird die Genauigkeit über der Trefferquote der Ergebnisse von Puttemans et al. dargestellt. Die Abkürzungen in der Legende entsprechen denen in den Überschriften in Abschnitt 3.3.2. (Bildquelle: [39])

3.3.3 Ergebnisse

In Abbildung 11 ist das Genauigkeit-Trefferquote-Diagramm für die einzelnen Algorithmen zu sehen. Die besten Ergebnisse konnten Puttemans et al. mit dem *Boosted-Cascade*-Algorithmus erzielen: eine Genauigkeit von 93 % bei einer Trefferquote von 80 %. Da dieser Algorithmus lediglich den blauen Farbkanal verwendet, haben sie

versucht, die anderen Farbkanäle über ein anschließendes Anwenden des *HSV+SVM*-Algorithmus mit einzubeziehen. Das hat die Ergebnisse nicht sonderlich verbessert, weswegen sie diesen Schritt wieder weglassen.

Der *MSER*-Algorithmus zeigte zwar eine gute Genauigkeit, aber eine unbrauchbare Trefferquote ($<$20 %).

Als Probleme ihrer Vorgehensweise nennen Puttemans et al. die schlechte Bodenauflösung der Datenquelle, dass die analysierte Region rein städtisch war und Ausreißer in der optischen Beschaffenheit von Solaranlagen (z.B. komplett schwarze Module).

Puttemans et al. schlagen vor, für die Aufgabenstellung *Convolutional Neural Networks* (vgl. Abschnitt 2.2.1) auszuprobieren.

3.4 Forschung von Yuan et al. (2016)

2016 haben sich Yuan et al. an der Lokalisierung von PV-Modulen anhand von Orthophotos und einem *Deep Convolutional Neural Network* versucht (vgl. [20]).

3.4.1 Datenquelle

Yuan et al. haben RGB-Orthophotos mit einer Bodenauflösung von 30 cm von Städten in den USA verwendet, die zwischen 2012 und 2013 aufgenommen wurden. Sie erwähnen, dass auch Infrarotbilder oder Orthophotos mit besserer Bodenauflösung möglich gewesen wären. Sie haben diese aber wegen der schlechteren Verfügbarkeit nicht verwendet.

Zum Trainieren des Netzes haben sie einen 12 km^2 großen Bereich gewählt, und diesen aufgeteilt in Bilder, die je einen Bereich von 75×75 cm abdecken (vergrößert auf 500×500 Pixel). Die Bilder haben sie so ausgewählt, dass 5,1 % davon keine PV-Module beinhalten. Diese Maßnahme soll False Positives eliminieren.

3.4.2 Algorithmus

Bei der verwendeten Bodenauflösung machen viele PV-Module weniger als 100 Pixel aus. Innerhalb des neuronalen Netzes wird zur Abstraktion die Auflösung weiter verringert, wodurch noch weniger Pixel pro PV-Modul zur Verfügung stehen. Deshalb haben

Yuan et al. die Eingabedaten vorab zweifach durch Interpolation vergrößert (Upsampling) und nach der Verarbeitung wieder entsprechend verkleinert (Downsampling). Das hat das Netz deutlich schneller lernen lassen.

Weil PV-Module flächenmäßig einen sehr kleinen Anteil am gesamten Datensatz ausmachen, haben Yuan et al. versucht, aussagekräftigere Annotationen zu verwenden. Anstatt jedem Pixel einen binären Wert zuzuordnen (PV-Modul oder nicht), verwenden sie eine Ganzzahl. Diese gibt die Entfernung des Pixels zum nächstgelegenen Rand eines PV-Moduls an. Positive Zahlen bedeuten, dass der Pixel innerhalb des PV-Moduls liegt, negative Zahlen stehen für außerhalb. („signed distance value“)

Yuan et al. haben sich für ein *Convolutional Neural Network* (CNN, vgl. Abschnitt 2.2.1) entschieden. Die Architektur dessen ist in Abbildung 13 dargestellt, welche ihrer Arbeit entnommen ist. Die gewählte Architektur basiert auf Yuans vorangegangener Forschung (vgl. [45]).

Die Eingabe für das CNN ist ein RGB-Bild von beliebiger Größe. Darauf werden in sieben Stufen Faltungsoperationen und teils Max-Pooling-Layer (vgl. [22]) angewendet. Vier der sieben Stufen werden, nach Anwendung von Upsampling, zu einem *Feature Stack* zusammengesetzt. Dadurch können Informationen von verschiedenen semantischen Abstraktionsniveaus abgegriffen werden. In diesem gibt es für jeden Pixel des Eingabebildes einen *Feature-Vektor*.

Mit einer letzten Faltungsoperation („single-layer perceptron classifier“) wird ein Ausgabebild erzeugt, das die halbe Größe des Eingabebildes hat (weil dieses zuvor vergrößert wurde). Die letzte Faltungsoperation generiert 128 Werte pro Pixel. Diese Werte werden per *Softmax* auf den Wertebereich $(0, 1)$ normiert, sodass sie in Summe 1 ergeben. Jede Wahrscheinlichkeit entspricht einer gerichteten Entfernung zum Rand eines PV-Moduls, entsprechend den *signed distance value*-Annotationen, siehe Abbildung 12. Die Entfernungen multipliziert mit den Wahrscheinlichkeiten ergeben in Summe die Ausgabewahrscheinlichkeit, ob der jeweilige Pixel einem PV-Modul entspricht.

Yuan et al. haben das CNN per *Stochastic Gradient Descent* für 140 Epochen trainiert.

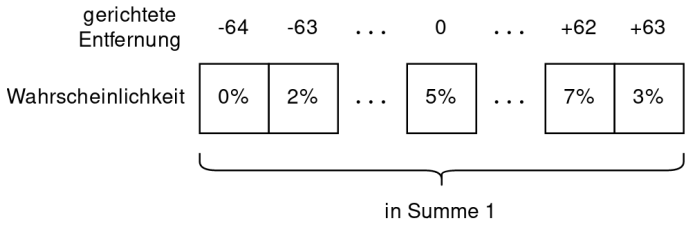

Abbildung 12: Dies ist ein Beispiel für einen von Yuan et al. verwendeten Vektor, der eine Folge von Wahrscheinlichkeiten enthält, die summiert 100 % ergeben. Für jede gerichtete Entfernung in Pixeln, die der betrachtete Pixel zum Rand eines PV-Moduls haben könnte, gibt es eine Wahrscheinlichkeit.

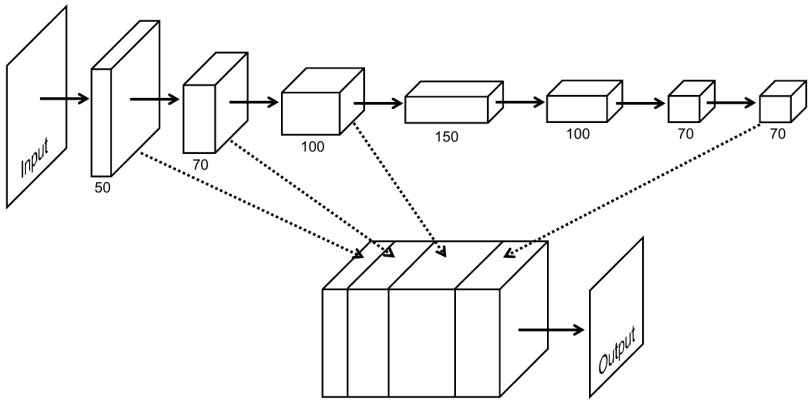

Abbildung 13: Architektur des CNNs von Yuan et al. Durchgezogene Pfeile stehen für Faltungsoperationen, gestrichelte Pfeile für Upsampling. Die Zahlen geben an, wie viele Filter bei der Faltungsoperation verwendet werden. (Bildquelle: [20])

3.4.3 Ergebnisse

Um das erstellte neuronale Netz zu Bewerten, bestimmen Yuan et al. die Genauigkeit und Trefferquote. Diese werden allgemein anhand dessen berechnet, wie viele Klassifizierungen positiv („PV-Modul“) oder negativ („kein PV-Modul“) und korrekt oder falsch sind. Wird diese Berechnung anhand einzelner Pixel durchgeführt, beeinflusse laut Yuan et al. ein einzelner Pixel die Ergebnisse unerwünscht stark. Das rühre daher, dass die PV-Module nur einen kleinen Anteil an den Bildern ausmachen. Deswegen ermitteln sie für jede Klassifizierung den zentralen Pixel und erweitern den annotierten („wahren“) Bereich je PV-Modul um einen Meter. Liegt ein zentraler Pixel innerhalb eines annotierten Bereiches, wird das als korrekte positive Klassifizierung gewertet. Diese Bewertung haben Yuan et al. für zwei Test-Bilder durchgeführt, die je einen Bereich von 108 km^2 abgedeckt haben: Mit dem Bild von *San Francisco* erreichten sie eine Genauigkeit von 85,5 % und eine Trefferquote von 87,3 %, bei *Boston* 81,2 % bzw. 84,0 %.

Die Genauigkeit konnten Yuan et al. um 3 % respektive 2 % verbessern, indem sie einige False Positives entfernten. Dafür glichen sie die Klassifizierungen mit Koordinaten von Straßen aus *OpenStreetMap* (vgl. [25]) ab. Alle als solche klassifizierten PV-Module, die sich mit Straßen überschnitten, wurden entfernt.

3.5 DeepSolar: Forschung von Yu et al. (2018)

In ihrer 2018 erstellen Arbeit haben Yu et al. unter anderem eine Software namens „DeepSolar“ entwickelt (siehe [19]). DeepSolar lokalisiert PV-Module anhand von Satellitenbildern.

3.5.1 Datenquelle

Die Satellitenbilder haben Yu et al. von der *Google Static Maps API* für die Region der gesamten USA bezogen. Diese Fotos basieren auf Satellitenbildern mit 15 cm Bodenauflösung, sind allerdings auf unterschiedliche Auflösungen umgerechnet. Die effektive Bodenauflösung variiert nach Region ungefähr zwischen 15 und 21 cm. Die Bilder werden jährlich aktualisiert – wann die von Yu et al. verwendeten Bilder aufgenommen

wurden, geben sie allerdings nicht an.

Unter Einsatz von Crowdsourcing (menschliche Helfer) und einem *Convolutional Neural Network* (CNN) haben Yu et al. einen Datensatz von über 470 000 annotierten Satellitenbildern erstellt. Dabei wurden die Bilder in „enthält PV-Modul“ und „enthält kein PV-Modul“ eingeteilt. Diesen Datensatz haben sie unterteilt in Trainings- (77,5 %), Validierungs- (2,7 %) und Test-Datensätze (19,8 %). Die Bilder im Test-Datensatz wurden zufällig aus 35 geographischen Regionen ausgewählt, wobei diese Regionen nicht in den Trainings-Daten enthalten sind. Außerdem wurden zu den Bildern im Test-Datensatz von Menschen pixelgenau die Standorte von PV-Modulen erfasst.

3.5.2 Algorithmus

Zum Klassifizieren der Satellitenbilder verwenden Yu et al. ein CNN. Dieses bestimmt zunächst nur, *ob* sich auf dem Bild ein PV-Modul befindet. Weil der Anteil der Bilder mit PV-Modul im Trainings-Datensatz sehr klein ist, passen sie die Loss-Funktion an: False-Positives werden stärker bestraft (vgl. „cost-sensitive learning“ [46]). Die Bilder werden zwecks Variation vor dem Trainieren zufällig in 90°-Schritten gedreht. Yu et al. haben sich einem bereits vortrainierten CNN namens *Inception-v3* (vgl. [47]) bedient. Mittels Transfer-Lernen (vgl. [48]) haben sie dieses CNN für ihre Zwecke angepasst: Der letzte Layer wurde mit zufälligen Werten initialisiert, bei allen anderen Layern stammen die initialen Werte aus Inception-v3. Für das anpassen der Werte verwenden sie einen *RMSProp optimizer* (vgl. [49]).

Die Fähigkeit, pixelgenau die PV-Module zu lokalisieren wird über ein weiteres, mit dem ersten verknüpften CNN geschaffen. Dieses wird *teilüberwacht* („semi-supervised“, vgl. [50]) trainiert, denn die Trainingsdaten sind nur pro Bild, nicht pro Pixel klassifiziert. Das CNN hat somit lediglich die pro-Bild-Klassifizierung, um zu lernen, Pixel zu klassifizieren. Jedem Pixel wird eine Wahrscheinlichkeit zugeordnet, ob es sich dabei um ein PV-Modul handelt. Über einen Schwellwert wird diese Wahrscheinlichkeit zu einer binären Aussage gemacht. Durch den pixelgenau annotierten Test-Datensatz kann das CNN dann bewertet werden. Yu et al. betonen, dass das *teilüberwachte* gegenüber *vollüberwachtem* Lernen die landesweite Anwendung erst möglich gemacht hat. Weil

dabei der Trainings-Datensatz nicht pixelgenau annotiert werden muss, erleichtert das die Datensatz-Erstellung wesentlich. Außerdem spart das teilüberwachte Lernen viel Rechenaufwand.

3.5.3 Ergebnisse

Yu et al. haben versucht, mit ihrer Software DeepSolar alle PV-Anlagen in den USA[13] zu lokalisieren. Sie haben alle Satellitenbilder zunächst auf urbane Gegenden und Gegenden ab einer bestimmten Helligkeit bei Nacht gefiltert, basierend auf Daten vom *U.S. Census Bureau* bzw. der *NASA*. Sie schätzen, damit 95 % aller Bilder mit PV-Modulen zu verarbeiten. Sie versprechen, die ausgelassenen Bilder nachträglich ebenfalls zu verarbeiten. Anhand ihrer Arbeit schätzen Yu et al., dass im untersuchten Gebiet auf 2,197 Millionen Bildern PV-Module zu erkennen sind. Über das entsprechende Gebäude oder den gleichen Standort zusammengefasst zählen sie 1,4702±0,0007 Millionen PV-Anlagen. Sie attestieren DeepSolar eine Genauigkeit von 93,1 % und eine Trefferquote von 88,5 % in Wohngebieten und 93,7 % respektive 90,5 % in nicht-Wohngebieten. Der gesamte Verlauf dieser Werte ist in Abbildung 14 zu finden.

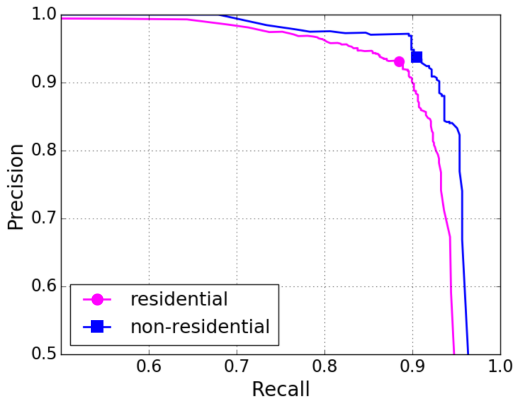

Abbildung 14: Hier wird die Genauigkeit über der Trefferquote der Ergebnisse von Yu et al. in bewohnten und nicht bewohnten Gegenden dargestellt. (vgl. Abschnitt 3.5, Bildquelle: [19])

Sie haben ihre detaillierten Ergebnisse und ihren Quellcode veröffentlicht.[14]

[13] „contiguous U.S.“

[14] vgl. `https://web.stanford.edu/group/deepsolar/home` (abgerufen am 23.05.2020)

3.6 Forschung von Castello et al. (2019)

Die bisher vorgestellte Forschung zur Lokalisierung von PV-Anlagen hat sich auf die USA und Belgien bezogen. Castello et al. haben in ihrer 2019 veröffentlichten Arbeit das gleiche Problemfeld für die Schweiz untersucht (vgl. [21]). Sie haben ebenfalls Orthophotos mit neuronalen Netzen analysiert.

3.6.1 Datenquelle

Castello et al. haben Orthophotos mit einer Bodenauflösung von 25 cm verwendet, die im Jahr 2013 aufgenommen und von dem *Swiss Federal Office of Topography* bereitgestellt wurden. Sie haben 700 250×250 Pixel-große Bilder mit händisch annotierten PV-Modulen erstellt. Zusammen mit 80 solchen Bildern ohne PV-Module haben sie diese zum Trainieren eines neuronalen Netzes verwendet. Um mehr Trainingsdaten zu bekommen haben sie die Bilder in 90°-Schritten rotiert und die Lichtverhältnisse („lighting“) variiert. Die geänderten Bilder haben sie während des Trainierens sukzessive den Trainingsdaten hinzugefügt.

3.6.2 Algorithmus

Als Architektur für ihr neuronales Netz haben sich Castello et al. für ein *U-Net* (vgl. [51]), welches ein populäres *Convolutional Neural Network* ist, entschieden. Dieses verdichtet die Eingabe zunächst durch *Max-Pooling*-Layer auf wenige, abstrahierte Informationen und überträgt diese dann per *Up-Sampling* bis zu einem Layer, der wieder die Größe des Eingabe-Layers hat. Als Loss-Funktion verwenden Castello et al. eine gewichtete, Pixel-basierte kategoriale Kreuzentropie-Funktion (vgl. [52]). Die gewählte Funktion gewichtet falsche positive Klassifizierungen fünffach stärker, weil in den Trainingsdaten die Pixel mit PV-Modul gegenüber denen ohne PV-Modul deutlich in der Unterzahl sind. Zur Optimierung des Loss verwenden Castello et al. *Adam* (vgl. [53]). Sie trainieren mit 32 Bilder-großen Batches und verwenden 20 % der Trainings-Bilder für die Validierungen. Sie trainieren ihr CNN für 75 Epochen, was sie als Optimum im Hinblick auf Overfitting beschreiben.

3.6.3 Ergebnisse

Weil die Pixel mit PV-Modul einen so kleinen Anteil am gesamten Datensatz haben, betonen Castello et al., dass der Jaccard-Index eine aussagekräftigere Metrik als die Korrektklassifikationsrate („accuracy“)[15] sei. Sie erzielen einen Jaccard-Index von 0,64 und eine Korrektklassifikationsrate von 94 %. Das Genauigkeits-Trefferquote-Diagramm ist in Abbildung 15 zu finden. Einige False Positives waren auf dem Boden befindliche Objekte, die durch ihr Aussehen und ihre Umgebung PV-Modulen ähnelten. Castello et al. vermuten, dass das durch mehr Bilder ohne PV-Module in den Trainingsdaten oder Up-Sampling der Trainingsdaten mittels Interpolation verbessert werden kann. Eine Kantendetektion vor Eingabe in das CNN könne ebenso helfen.

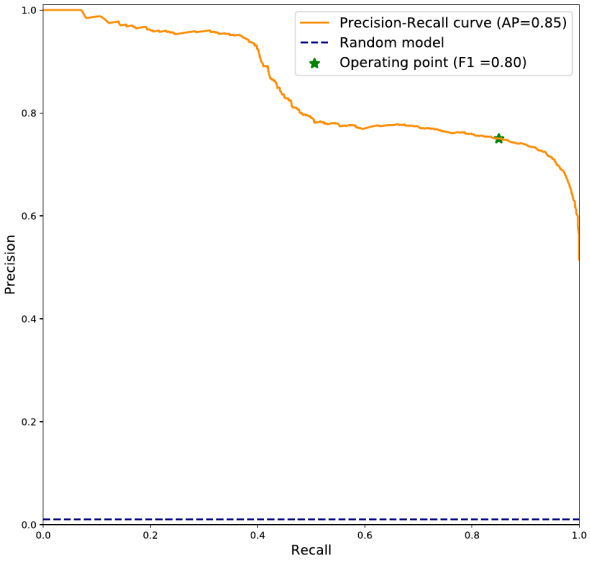

Abbildung 15: Hier wird die Genauigkeit über der Trefferquote der Ergebnisse von Castello et al. dargestellt. (vgl. Abschnitt 3.6, Bildquelle: [21])

3.7 Zusammenfassung des Standes der Forschung

Wie in den vorangegangenen Abschnitten zusammengefasst wurde, sind wir nicht die ersten, die versuchen, PV-Anlagen anhand von Orthophotos zu lokalisieren. Eine diverse Sammlung an Algorithmen und Datenquellen wurde verwendet, darunter sehr vielversprechende Ansätze. *Convolutional Neural Networks* sind eine beliebte Form des maschinellen Lernens – vier der vorgestellten Arbeiten haben sie verwendet, und Puttemans et al. schlagen sie für zukünftige Forschung vor. Für die verwendeten CNNs

[15] Die Korrektklassifikationsrate ist der Anteil der korrekten an allen Klassifizierungen.

wurden allesamt bereits bestehende Architekturen oder sogar Netze (trainierte Architekturen für Transfer-Lernen) genutzt.

Die Diversität macht es schwierig, die verschiedenen Ansätze zu vergleichen. Eine gängige Metrik sind Genauigkeits-Trefferquote-Diagramme (GT). Allerdings unterscheiden sich diese darin, *wovon* die jeweiligen Werte berechnet wurden. GT beziehen sich immer auf die Anzahlen der Elementen, die positiv oder negativ klassifiziert wurden, und über einen entsprechenden Datensatz validiert werden. Die Elemente sind z.B. bei Malof et al. mal einzelne Pixel (vgl. Abschnitt 3.2.3), mal variable Regionen von Pixeln (vgl. Abschnitt 3.2.1) und mal Regionen fixer Größe (vgl. Abschnitt 3.2.4). Die Elemente bei Puttemans et al. sind Pixel, allerdings mit der halben Auflösung der Eingabedaten. Bei den anderen vorgestellten Arbeiten findet man ähnliche Vielfalt.

Für die Überprüfung und für gewöhnlich auch für das Trainieren der vorgestellten Algorithmen werden Orthophotos benötigt, zu denen pixelgenau vermerkt ist, wo sich PV-Module befinden. Diese werden händisch erstellt, was mit wachsender Anzahl sehr aufwändig wird. Yu et al. haben versucht, diesen Aufwand zu verringern. Mittels teilüberwachtem Lernen konnten sie ein CNN dazu bringen, PV-Module pixelgenau zu klassifizieren, wobei die Trainingsdaten nur pro Bild in „enthält PV-Modul“ und „ohne PV-Modul“ klassifiziert werden mussten. Für die Validierung werden weiterhin händisch erstellte, pixelgenaue Annotationen benötigt.

Die Menge, Bodenauflösung und geographische Region der verwendeten Orthophotos variiert, was in Tabelle 1 veranschaulicht wird. Das hemmt die Vergleichbarkeit der Arbeiten. Zu zwei der betrachteten Arbeiten wurde der verwendete Datensatz veröffentlicht.

Aus den Arbeiten geht hervor, dass die Bodenauflösung der verfügbaren Orthophotos eine Schwierigkeit darstellt: Ein PV-Modul wird schlicht durch ziemlich wenig Pixel abgebildet. In anderen Anwendungsgebieten von maschinellem Lernen gibt es hingegen meist genug Datenpunkte, sodass die in den Algorithmen zur Abstraktion vorgenommene Aggregierung der Datenpunkte genug Auflösung übrig lässt – in diesem Fall haben sich die vorgestellten Forscher jedoch verschiedene Abhilfen überlegt:

- Puttemans et al. und Yuan et al. vergrößern die Eingabedaten durch Interpolation, Castello et al. schlagen dies in ihrem Ausblick vor.

	Bodenauflösung	Fläche	Region	veröffentlicht
Malof et al. 2015	30 cm	0,196 km^2	USA	nein
Malof et al. 2016	30 cm	135 km^2	USA	ja: [1]
Puttemans et al. 2016	25 cm	4 km^2	Belgien	ja: [40]
Yuan et al. 2016	30 cm	12 km^2	USA	nein
Yu et al. 2018	15-21 cm	>50 Städte/Dörfer	USA	nein
Castello et al. 2019	25 cm	48,75 km^2	Schweiz	nein

Tabelle 1: Vergleich der in vorgestellten Arbeiten zum Trainieren der Algorithmen verwendeten Orthophotos. Yu et al. geben keine exakte Fläche an, die sie mit ihren Trainingsdaten abgedeckt haben.

- Yuan et al. verwenden statt binären Annotationen der Trainingsdaten *signed-distance-value*-Annotationen.
- Malof et al., Yu et al. und Castello et al. rotieren die Bilder des Trainingsdatensatzes in mehreren Winkeln, um mehr Variation zu bekommen. Castello et al. haben außerdem die Lichtverhältnisse der Bilder variiert.

Die Pixel, die PV-Modulen entsprechen, haben üblicherweise einen kleinen Anteil an allen Pixeln der betrachteten Landschaft. Malof et al. betrachten deswegen in einer ihrer Arbeiten (vgl. Abschnitt 3.2.1) nur Hausdächer. Yu et al. und Castello et al. gewichten deswegen falsche positive Klassifizierungen in der Lern-Funktion ihres CNNs strenger.

Es gibt außerdem Ideen, die Genauigkeit der Lokalisierung mithilfe von öffentlichen Kartendaten zu verbessern:

- Yuan et al. verwenden OpenStreetMap, um falsche positive Klassifizierungen daran zu filtern, dass sie auf Straßen vermutet wurden.
- Killinger verwendet OpenStreetMap, um Modulorientierungen zu bestimmen.
- Yu et al. verwenden Daten von Behörden der USA, um das zu analysierende Gebiet auf urbane Gegenden und Gegenden mit bestimmter Helligkeit bei Nacht zu reduzieren.

Zur Verbesserung der Pixel-basierten Klassifizierung wurden verschiedene ergänzende Algorithmen evaluiert. Beispielsweise haben Malof et al. (vgl. Abschnitt 3.2.2) und Puttemans et al. die morphologischen Operationen *Closing* und *Opening* verwendet.

Keine der zusammengefassten Arbeiten hat die optische Ähnlichkeit von PV-Modulen und Solarkollektoren (vgl. Abschnitt 1.1) erwähnt, obwohl in der Regel speziell von PV die Rede ist. Ob und wie dieser Unterschied berücksichtigt wurde, ist unklar. Puttemans et al. betonen, dass PV-Module hauptsächlich blau aussehen und verwenden deswegen teilweise sogar nur den blauen Farbkanal der Orthophotos. Allerdings räumen sie ein, dass es vom durchschnittlichen Aussehen abweichende Anlagen gibt, die ihr Algorithmus nicht erkennt.

4 Evaluation möglicher Datenquellen

In diesem Abschnitt wird erörtert, welche Datenquellen wir für die Verwendung mit unserem Datensatz heranziehen. Davon ist abhängig, wie gut und aktuell wir PV-Anlagen lokalisieren können. Wir beschränken uns (bis auf beschriebene Ausnahmen) auf Datensätze zu Landschaften in Deutschland.

4.1 Marktstammdatenregister

Die Bundesnetzagentur führt ein Marktstammdatenregister[16] (kurz MaStR) genanntes Register, in dem unter anderem PV-Anlagen eingetragen werden müssen. Gemäß §5 Marktstammdatenregisterverordnung (kurz MaStRV) müssen PV-Anlagen („Stromerzeugungseinheiten“) registriert werden, wenn sie an ein Stromnetz angeschlossen sind oder werden sollen. Die MaStRV schreibt außerdem vor, dass die registrierten Informationen zu PV-Anlagen veröffentlicht werden müssen. Zu den erfassten Daten zählen unter anderem die Koordinaten und Orientierungen von PV-Modulen.

Allerdings werden genaue Standortdaten zu Solaranlagen mit einer Leistung höchstens 30 kW nicht veröffentlicht. Deshalb sind diese Daten nur zu einer Promille aller eingetragenen PV-Anlagen veröffentlicht.[17] Ohne die Standorte der PV-Module zu kennen, nützen uns auch die Modulorientierungen nichts. Daher verwenden wir statt dem MaStR die nachfolgend beschriebenen Orthophotos als Datenquelle.

4.2 Mögliche Quellen für Orthophotos

Das *Bundesamt für Kartographie und Geodäsie* (BKG) bietet digitale Orthophotos mit einer Bodenauflösung von bis zu 20 cm für die gesamte Fläche Deutschlands an (vgl.

[16] vgl. `https://www.marktstammdatenregister.de/MaStR/` (abgerufen am 27.05.2020)

[17] Der Anteil von PV-Anlagen mit öffentlichen Standort wurde am 23.05.2020 aus der Anzahl der Stromerzeugungseinheiten in den öffentlichen Daten des Marktstammdatenregisters ermittelt. Die für den Datensatz verwendeten Filter waren: „Anzahl der Solar-Module“ größer 0, „Betriebs-Status“ entspricht „In Betrieb“ und „Energieträger“ entspricht „Solare Strahlungsenergie“. Entsprechend der gesetzlichen Regelung gab die Hinzunahme des Filters „Bruttoleistung der Einheit“ größer als 30 000 Aufschluss über die PV-Anlagen mit öffentlichem Standort. (Datenquelle: `https://www.marktstammdatenregister.de/MaStR/Einheit/Einheiten/OeffentlicheEinheitenuebersicht`)

[54]). Diese Daten werden von den Vermessungsverwaltungen der Bundesländer erhoben und vom BKG als einheitliche Sammlung zur Verfügung gestellt. Die Aufnahmezeiten der Bilder variieren und sind bis maximal 6 Jahre alt.[18] Es stehen verschiedene Koordinatensysteme für die Georeferenzierung zur Auswahl, mit einer Lagegenauigkeit von $\pm 40\,\text{cm}$.

Allerdings berechnet das BKG einen sechsstelligen Betrag für den gesamten Datensatz. Da uns so viel Geld nicht zur Verfügung steht, und wir uns auf frei zugängliche Datensätze beschränken, kam dieser nicht in Frage.

Die Bundesländer bieten ihre Datensätze auch selbst an, und unterscheiden sich in Qualität, technischem Format und Lizenzbestimmungen. Tabelle 2 schlüsselt die von den Bundesländern veröffentlichten Datensätze nach Bodenauflösung und Nutzungsbedingungen auf.

Die Recherche offenbart, dass Nordrhein-Westfalen als einziges Bundesland öffentlich und kostenfrei Orthophotos mit einer Bodenauflösung von 10 cm anbietet. Von Berlin, Brandenburg, Hamburg und Sachsen gibt es solche mit einer Bodenauflösung von 20 cm. Ungenauere Orthophotos halten wir aufgrund der Größe von PV-Modulen für unsere Arbeit für ungeeignet.

4.2.1 Verwendete Quellen

Aufgrund der im vorherigen Abschnitt erläuterten Recherche haben wir 2018 aufgenommene Orthophotos von NRW mit 10 cm Bodenauflösung verwendet (vgl. [10]). Damit haben wir eine bessere Bodenauflösung als die in Abschnitt 3 betrachtete Literatur

[18] vgl. `https://sg.geodatenzentrum.de/web_public/gdz/aktualitaet/dop.html` (abgerufen am 10.03.2020)

[19] Für wissenschaftliche Abschlussarbeiten stellt das *Landesamt für Geoinformation und Landentwicklung Baden-Württemberg* digitale Ortophotos kostenfrei zur Verfügung, begrenzt auf eine Datenart (Bodenauflösung) und eine Fläche von $1\,\text{km}^2$.

[20] vgl. `https://www.lgl-bw.de/export/sites/lgl/unsere-themen/Produkte/Galerien/Dokumente/VwVNutzGeo_2009.pdf` (abgerufen am 16.03.2020)

[21] vgl. `https://www.ldbv.bayern.de/file/pdf/1269/Preisliste_aktuell.pdf#page=9&view=Fit` (abgerufen am 16.03.2020)

[22] vgl. `https://www.schleswig-holstein.de/DE/Landesregierung/LVERMGEOSH/LVermGeoShBilderPdf/pdfGesetzErlassVerordnung/vermEgo2018.pdf?__blob=publicationFile&v=4` (abgerufen am 16.03.2020)

Bundesland	frei	eingeschränkt	Quellen
BW	-[19]	10 cm kostenpflichtig	[55] und VwVNutzGeo[20]
BY	200 cm	20 cm kostenpflichtig	[56] und GebPL[21]
BE	20 cm	10 cm kostenpflichtig	[57]
BB			
HB	-	10 cm auf Anfrage	[58]
HH	20 cm	-	[59], [60]
HE	-	20 cm (teilweise 10 cm) kostenpflichtig	[14], [61], [62]
MV	-	10 cm kostenpflichtig und benötigt Darlegung eines berechtigten Interesses, 20cm kostenpflichtig	[9]
NI	-	20 cm kostenpflichtig	[15], [63]
NW	10 cm	-	[10], [11]
RP	40 cm	20 cm kostenpflichtig	[16], [64], [65]
SL	-	20 cm mit individuellem Vertrag	[66]
SN	20 cm	-	[67]–[69]
ST	100 cm	10 cm	[12]
SH	-	20 cm kostenpflichtig	[70] und VermEgO[22]
TH	200 cm	20 cm kostenpflichtig	[71], [72]

Tabelle 2: Jedes Bundesland bietet Orthophotos an, allerdings mit unterschiedlichen Bodenauflösungen und Konditionen. Diese Tabelle schlüsselt die genauste angebotene Bodenauflösung von Orthophotos nach Bundesland auf. Die Spalte „frei“ nennt dabei die genauste Bodenauflösung, die öffentlich und kostenfrei verfügbar ist. Teilweise sind keine Orthophotos öffentlich und kostenfrei erhältlich. Die Spalte „eingeschränkt“ nennt ggf. genauere angebotene Bodenauflösungen, die allerdings jeweils genannten Einschränkungen unterliegen. Wir verzichten hier auf eine Nennung der Aktualität der Daten, weil sich diese nicht es sich nicht einheitlich pro Bundesland beantworten lässt.

zur Verfügung (vgl. Tabelle 1). Die Orthophotos sind im Format *TrueDOP* und haben RGBI-Kanäle, wovon wir jedoch nur die RGB-Kanäle nutzen.

Da wir zur Erstellung von Trainingsdaten nicht das gesamte Bundesland händisch annotieren wollten, haben wir uns auf die 71,37 km^2 große Fläche der Stadt Verl, Kreis Gütersloh, Postleitzahl 33415, beschränkt. Wir haben dieses Postleitzahlgebiet deswegen ausgewählt, weil es die meisten PV-Module pro Fläche und die meisten PV-Module pro Einwohner enthält. Das haben wir anhand von Daten aus dem MaStR (vgl. [73]) und von dem Statistischen Bundesamt (vgl. [74]) berechnet.

Der Einfachkeit halber erweitern wir die betrachtete Region von den Stadtgrenzen auf ein umschließendes Rechteck, das durch die WGS84-Koordinaten[23] 51,924503° N 8,441370° E und 51,829527° N 8,609869° E begrenzt ist. Dieses Rechteck ist etwa 122 km^2 groß und wird im Nachfolgenden als *NRW-Rechteck* bezeichnet.

Kalifornien Außerdem verwenden wir den von Malof et al. veröffentlichten Datensatz (vgl. [1] und Abschnitt 3.2). Dieser enthält Orthophotos mit annotierten Standorten von Solaranlagen zu den kalifornischen Städten *Fresno*, *Stockton*, *Oxnard*, und *Modesto* und deckt damit eine Fläche von 135 km^2 ab.

4.3 Georeferenzierte Umringpolygone von Gebäudedächern

Für den in Abschnitt 5.3 beschriebenen Zuschnitt der Orthophotos auf die Flächen von Gebäudedächern benötigen wir georeferenzierte Umringpolygone von Gebäudedächern. Wir verwenden zur Vereinfachung und wegen der Verfügbarkeit von Datensätzen die Annahme, dass ein Umringpolygon eines Gebäudes genau einem Umringpolygon eines Daches entspricht. Für georeferenzierte Umringpolygone von Gebäuden (nachfolgend „Umringe“) gibt es verschiedene mögliche Datenquellen, von denen einige im Folgenden vorgestellt werden.

Das *Bundesamt für Kartographie und Geodäsie* verwaltet einen Datensatz an Umringen (siehe [75]). Weil dieser Datensatz aus vollständigen, präzisen, amtlichen Datenerhebungen besteht, wäre er ideal für unseren Einsatzzweck. Allerdings können wir

[23] OSM verwendet das *World Geodetic System 1984* (WGS84) als geodätisches Referenzsystem, vgl. `https://josm.openstreetmap.de/wiki/Help/Concepts/Coordinates` (abgerufen am 09.03.2020).

ihn nicht nutzen, weil dieser nur für in *§4 V GeoBund* spezifizierte Personen (z.B. Bundesbehörden) zugänglich ist.

Eine Alternative ist die öffentlich zugängliche Datenbank *OpenStreetMap* (OSM, vgl. [25]), welche unter anderem Umringe enthält. OSM stellt eine HTTP-API bereit, über die die Geoinformationen im XML-Format abgefragt werden können. Die Daten in OSM stammen, statt von einer amtlichen Quelle, von Freiwilligen, und sind deshalb nicht notwendigerweise korrekt oder vollständig. Verschiedene Studien haben sich mit der Qualität der Daten in OSM beschäftigt.

2013 haben Fan et al. (vgl. [76]) die Qualität von Umringen in OSM im Vergleich zum *Amtlichen Topographisch-Kartographischen Informationssystem* (ATKIS) untersucht. Bei letzterem handelt es sich um einen von Behörden gepflegten Datensatz für gesamt Deutschland, welcher die bereits erwähnten amtlichen Umringe enthält. Fan et al. haben den Vergleich am Beispiel der Stadt München im Hinblick auf Vollständigkeit, semantische Korrektheit, Positions- und Form-Korrektheit durchgeführt. Sie haben herausgefunden, dass die Positionen der Umringe durchschnittlich um 4 Meter falsch liegen. Das führen sie auf die unpräzisen Luftbilder von *Bing Maps* zurück, welche für die Erstellung der Umringe in OSM verwendet wurden. Wegen dieser Datenquelle fehlten in OSM außerdem einige Umringe zu Gebäuden, die in den Luftbildern nicht zu erkennen waren. Andererseits fanden Fan et al. auch viele Umringe von Neubauten, die nicht in ATKIS enthalten waren. Fast alle Umringe in ATKIS waren auch in OSM zu finden. Fan et al. bemängeln fehlende Attribute mit weiteren Informationen an den Umringen und weniger Detailreichtum der Formen im Gegensatz zu ATKIS. Sie zeigen sich jedoch optimistisch, dass durch die vielen an OSM beteiligten Freiwilligen die Datenqualität dessen bald verbessert wird.

Da die Arbeit von Fan et al. nun schon über 6 Jahre alt ist, ist davon auszugehen, dass sich die Datenqualität in OSM inzwischen verändert hat. 2015 haben Arsanjani et al. (vgl. [77]) die in OSM eingetragene Landnutzung an Daten aus dem *Copernicus* Erdbeobachtungsprogramm gemessen, welches von der *Europäischen Kommission* und der *Europäischen Weltraumorganisation* gegründet wurde. Sie kommen zu dem Schluss, dass OSM für die untersuchten Aspekte eine gute Datenquelle ist.

OSM wurde bereits im Kontext der Positionsbestimmung von PV-Anlagen heran-

gezogen, vgl. dazu beispielsweise Abschnitt 3.1.1 und Abschnitt 3.4.

Deswegen verwenden wir für unsere Arbeit für die Region Deutschland OSM als Datenquelle für Umringe und nehmen an, dass dieser Datensatz vollständig und korrekt ist.

4.3.1 Verwendete Quellen

Die Umringe für das *NRW-Rechteck* (vgl. Abschnitt 4.2.1) beziehen wir aus *OpenStreetMap* (OSM) (vgl. Abschnitt 4.3).

Wir haben uns den Datensatz zu NRW von der *Geofabrik GmbH* heruntergeladen.[24] Dieser Datensatz enthält unter anderem alle bis einschließlich zum 19.05.2020 in OSM gespeicherten Umringe. Mit dem Kommandozeilenwerkzeug Osmosis[25] filtern wir diesen Datensatz auf Umringe, die im NRW-Rechteck liegen. Alle *Ways* (Polygone), die das Attribut `building`[26] gesetzt haben, verwenden wir als Umring.

Unzureichende Vollständigkeit von Umringen in Kalifornien Wir verwenden nicht nur Orthophotos von Deutschland, sondern auch von Kalifornien (vgl. Abschnitt 4.2.1). Diese wollen wir ebenfalls auf Umringe zuschneiden.

Zum Zeitpunkt der Bearbeitung (März 2020) waren Umringe in OSM zwar für deutsche Städte in hinreichender Quantität eingezeichnet – für die kalifornischen Orte galt dies jedoch nicht. Abbildung 16 veranschaulicht exemplarisch, dass in den fraglichen Orten viele Gebäude nicht als Umring eingetragen sind. Deswegen brauchten wir eine andere Datenquelle für Umringe.

Die *Microsoft Corporation* hat auf *GitHub* einen Datensatz an Umringen für die gesamte USA veröffentlicht (vgl. [78]).[27] Die Umringe wurden mit einem neuronalen Netz anhand von Satellitenbildern mit einer Bodenauflösung von einem Fuß pro Pixel berechnet. Sie attestieren dem Algorithmus, mit dem sie den Datensatz erstellt haben,

[24] vgl. `http://download.geofabrik.de/europe/germany/nordrhein-westfalen.html` (abgerufen am 29.05.2020)

[25] vgl. `https://wiki.openstreetmap.org/wiki/Osmosis` (abgerufen am 29.05.2020)

[26] vgl. `https://wiki.openstreetmap.org/wiki/Buildings` (abgerufen am 29.05.2020)

[27] Mit gleichen Methoden hat Micosoft auch Datensätze für Kanada, Uganda und Tanzania, jedoch nicht für Deutschland, erstellt. (Stand: März 2020)

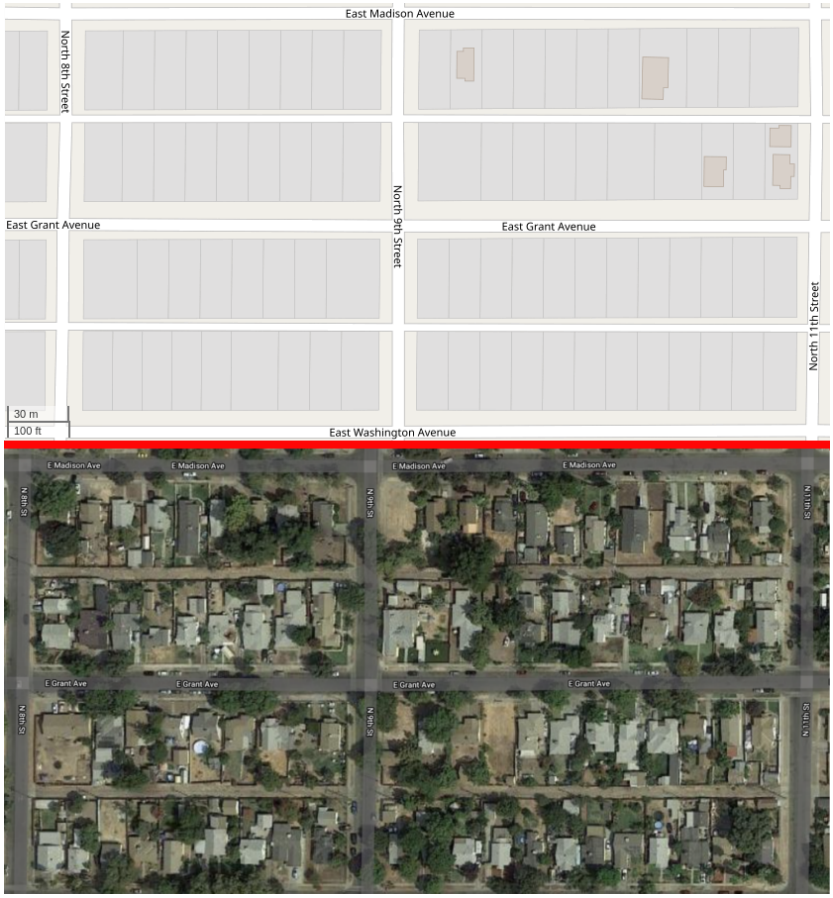

Abbildung 16: Dieser beispielhafte Kartenausschnitt von der kalifornischen Stadt *Fresno* zeigt die unvollständige Eintragung von Gebäuden in *OpenStreetMap (OSM).* Die obere Bildhälfte zeigt den Kartenausschnitt in OSM, die untere selbigen in *Google Maps* (beide abgerufen am 24.03.20). Die braunen Polygone in OSM entsprechen Gebäuden. Die grauen Rechtecke entsprechen vermutlich Grundstücken. Für viele in Google Maps erkennbare Gebäudedächer ist in OSM kein Gebäude eingezeichnet.

eine Genauigkeit von 99,3 % und eine Trefferquote von 93,5 %. Für Kalifornien sind knapp 11 Millionen Umringe im Datensatz verzeichnet. Aufgrund der guten Zahlen und der freien Verfügbarkeit haben wir das Zuschneiden der kalifornischen Orthophotos mit den Umringen aus diesem Datensatz gemacht.

5 Aufbereitung der Eingabedaten

Für die in Abschnitt 6 beschriebene Klassifizierung verwenden wir zwei Datensätze: Kalifornien und NRW. Diese Umfassen Orthophotos und händische Annotationen, die angeben, wo sich darin Solaranlagen befinden. Weil wir uns auf die Lokalisierung von Aufdachanlagen beschränken, schneiden wir die Orthophotos so zu, dass nicht viel mehr als einzelne Gebäudedächer auf den Orthophotos zu sehen sind. Wie diese Datensätze erstellt wurden, wird in den folgenden Unterabschnitten umrissen.

5.1 Datensatz NRW

In Abschnitt 4.2.1 haben wir das ausgewählte *NRW-Rechteck* beschrieben, welches die Fläche beschreibt, in der wir PV-Anlagen lokalisieren wollen. Die Orthophotos dazu beziehen wir über eine API von *Geobasis NRW* (vgl. [10]). Diese verwendet das *UTM* Koordinatenreferenzsystem (genauer: *EPSG 25832*), in das wir die *WGS84*-Koordinaten des Rechtecks konvertieren. Über die API bekommen wir für diese Eingabe 143 *JPEG 2000*[28]-Bilder mit zugehörigen *Wordfiles*[29]. Erstere sind 10000×10000 Pixel große Orthophotos, letztere enthalten Informationen für die Georeferenzierung. Wegen der Bodenauflösung von 10 cm entsprechen diese Bilder zusammen einer Fläche von 143 km^2. Das ist etwa 17 % größer als das ausgewählte Rechteck und etwa 100 % größer als das ausgewählte Postleitzahlgebiet.

Die Orthophotos konvertieren wir von JPEG 2000 zu PNG, weil das unsere Implementierung erleichtert hat.

Diesen Datensatz haben wir mit dem in Abschnitt 5.3 beschriebenen Algorithmus in 36 516 Dachbilder umgewandelt.

2000 dieser Dachbilder (zufällige Auswahl aus Zeitgründen) haben wir manuell gesichtet und dabei 143 Solaranlagen gefunden. Nach entfernen von Dachbildern, deren Inhalt sich nicht identifizieren ließ, haben wir 1638 Dachbilder ohne Solaranlagen. Zu den Dachbildern mit Solaranlagen haben wir mithilfe der Software *labelme* (vgl. [79])

[28] vgl. ISO 15444

[29] vgl. `https://desktop.arcgis.com/de/arcmap/latest/manage-data/raster-and-images/world-files-for-raster-datasets.htm` (abgerufen am 28.05.2020)

pixelgenau annotiert, wo sich Solaranlagen befinden. Diesen Datensatz haben wir auf *GitLab* veröffentlicht, siehe [2].

5.2 Datensatz Kalifornien

Die in genannten Orthophotos, die 135 km^2 Fläche in Kalifornien abdecken, haben wir mit den in Abschnitt 4.3.1 beschriebenen Umringen und dem in Abschnitt 5.3 beschriebenen Algorithmus in 341 379 Dachbilder umgewandelt. In 3,81 % (12 540) dieser Dachbilder war mindestens ein Pixel als einer Solaranlage zugehörig annotiert. Die annotierten Solaranlagen waren im Datensatz als Umringpolygone mit WGS84-Koordinaten eingetragen. Die Orthophotos lagen im TIFF-Format vor und wurden Umwandeln in Dachbilder zu JPEG konvertiert.

5.3 Zuschneiden auf Gebäudedächer

Da wir uns auf die Lokalisierung von Aufdachanlagen beschränken, wollen wir unseren Datensatz vor der eigentlichen Verarbeitung auf Gebäudedächer filtern. Dafür schneiden wir die Orthophotos so zu, dass pro Bild nicht viel mehr als ein Gebäudedach zu sehen ist. Die zugeschnittenen Bilder nennen wir *Dachbilder*. Bei der Gelegenheit erstellen wir zusätzlich passende binäre Rastergrafiken, die für jeden Pixel des Dachbildes angeben, ob dieser einem Solarmodul entspricht. Optional „schwärzen“ wir alle Pixel, die nicht dem Gebäudedach entsprechen, indem wir ihre Werte auf Null setzen.

5.3.1 Algorithmus

Für das Zuschneiden benötigen wir Orthophotos, Koordinaten von Solarmodulen (Annotationen) und Umringe (georeferenzierte Umringpolygone von Gebäudedächern).

Wir iterieren über die Orthophotos.

Zu jedem Orthophoto erstellen wir eine Rastergrafik, die für jeden Pixel angibt, ob dieser einem Solarmodul entspricht. Dafür erstellen wir zunächst eine mit Null initialisierte Rastergrafik der Größe des Orthophotos im PBM-Format[30]. Die WGS84-

[30] *Portable-Bitmap*-Standard, vgl. `http://netpbm.sourceforge.net/doc/pbm.html` (abgerufen am

Koordinaten aller Solarmodule rechnen wir in Pixel-Koordinaten relativ zum Orthophoto um. Überlappt eine Annotation mit dem Orthophoto, wird die Annotation ausfüllend mit dem Wert Eins in die PBM-Rastergrafik eingezeichnet. Sind keine Solarmodule im Bereich des Orthophotos bekannt, enthält die PBM-Rastergrafik nur Nullen.

Dann iterieren für das ausgewählte Orthophoto über alle Umringe.

Ausschneiden Von den Pixel-Koordinaten des Umrings bestimmen wir ein umschließendes Rechteck und erweitern dieses pro Rand um einen Puffer von 5 Pixeln (Erklärung folgt). Dazu siehe Algorithmus 1. Überlappt dieses Rechteck mit dem Orthophoto, wird ein Dachbild, bestehend aus der Überlappung, erstellt (vgl. Algorithmus 2). Der Teil dieser Überlappung wird sowohl aus dem Orthophoto als auch aus der PBM-Rastergrafik in je neue Dateien kopiert, wodurch ein Dachbild mit passender Annotation entsteht.

Der 5 Pixel breite Puffer umfasst Pixel, die zwar nicht klassifiziert werden, aber für die Klassifizierung von anderen Pixeln benötigt werden. Um den Feature-Vektor eines Pixels zu berechnen, müssen auch die Pixel in der Umgebung dessen betrachtet werden (vgl. Abschnitt 6.1). Die Umgebung von eigentlich ganz am Rand gelegenen Pixeln wird durch diesen Puffer zugänglich gemacht.

Schwärzen Soll das Dachbild *geschwärzt* werden, werden die folgenden Schritte angewendet (andernfalls ist das Dachbild fertig).

Die Koordinaten des Umrings werden auf Pixel-Koordinaten relativ zum Dachbild umgerechnet. Das Polygon in Pixel-Koordinaten skalieren wir so, dass es in seiner gesamten Höhe und Breite um je 30 Pixel wächst (vgl. Algorithmus 3). Dadurch schaffen wir einen Fehlertoleranzbereich für die aufgrund von Ungenauigkeiten der Datenquelle entstehenden Abweichungen der Georeferenzierung von Orthophotos im Vergleich zu Umringen.

Aus dem Umring wird eine Maske (*numpy*-Array, vgl. [81]) erstellt, die die Maße und Farbkanäle des Dachbildes hat. Alle Pixel (in allen Farbkanälen) der Maske werden mit Null initialisiert. Alle Pixel der Maske, die innerhalb oder auf dem Rand des Polygons liegen, werden auf Eins gesetzt. Die Maske wird dann Element-weise auf das Dachbild

01.05.2020)

Algorithmus 1 Mit diesen Algorithmus bestimmen wir aus einem Orthophoto (Parameter `self`) und einem Umring (Parameter `building`) den für ein Dachbild zu verwendenden Bereich in Pixel relativ zum Orthophoto. Der in Abschnitt 5.3.1 erläuterte Puffer steht hier in der Variable `self.cut_out_margin`. Die Variable `cv2` referenziert die *OpenCV*-Bibliothek (vgl. [80]), `np` die *NumPy*-Bibliothek (vgl. [81], [82]). Ausgabe des Algorithmus sind die Pixel-Koordinaten der oberen linken Ecke des auszuschneidenen Rechtecks und die Breite und Höhe dessen.

```
def get_bounding_box_for_building(self, building: CoordStructure) -> (int, int, int, int):
    px_coords = list(map(self.convert_coord_tuple_to_px, building))
    x, y, w, h = cv2.boundingRect(np.array(px_coords))
    x, w = add_and_center(x, w, self.cut_out_margin)
    y, h = add_and_center(y, h, self.cut_out_margin)

    # If not the whole building is outside: cut it out
    if (x < self.image.width and x + w > 0) and (y < self.image.height and y + h > 0):
        # Normalizing values into the picture
        if x < 0:
            w += x
            x = 0
        if y < 0:
            h += y
            y = 0
        if y + h > self.image.height:
            h -= (y + h) - self.image.height
        if x + w > self.image.width:
            w -= (x + w) - self.image.width
        # After the normalization w < width and h < height
        # x, y > 0
        # As well as the whole rect is within the picture

        # Normalizing to equal length of width and height
        if w > h:
            y -= int((w - h) / 2)
            h = w
            if y < 0:
                y = 0
            if y + h > self.image.height:
                y -= (y + h) - self.image.height
        elif h > w:
            x -= int((h - w) / 2)
            w = h
            if x < 0:
                x = 0
            if x + w > self.image.width:
                x -= (x + w) - self.image.width
        return x, y, w, h
    return None
```

Algorithmus 2 Dieser Algorithmus bekommt die von Algorithmus 1 berechneten Werte und ein Orthophoto als Parameter. Die Variable `np` referenziert die *NumPy*-Bibliothek (vgl. [81], [82]). Ausgabe ist ein neues Dachbild.

```
def cut_out_rect(self, x: int, y: int, w: int, h: int, image: np.ndarray = None) -> np.ndarray:
    if image is None:
        image = self.image
    if len(image.shape) == 3:
        cropped = image[:, y:y + h, x:x + w].copy()
    elif len(image.shape) == 2:
        cropped = image[y:y + h, x:x + w].copy()
    else:
        raise Exception('Wrong image dimension count: {}'.format(len(image.shape)))
    return cropped
```

Algorithmus 3 Diese Methode skaliert ein durch Pixel-Koordinaten gegebenes Polygon (erster Parameter) um einige Pixel größer (zweiter Parameter). Hierbei wird die *Shapely*-Bibliothek (vgl. [83]) verwendet. Ausgabe ist das skalierte Polygon.

```
import shapely.geometry as sg
from shapely.affinity import scale

def scale_polygon(polygon: PxCoordStructure, additional_pixels: int = 30):
    shapely_polygon = sg.Polygon(polygon)
    xs = [x for x, y in polygon]
    ys = [y for x, y in polygon]
    min_x = min(xs)
    min_y = min(ys)
    max_x = max(xs)
    max_y = max(ys)
    diff_x = max_x - min_x
    diff_x = 1 if diff_x == 0 else diff_x
    diff_y = max_y - min_y
    diff_y = 1 if diff_y == 0 else diff_y
    scale_x = additional_pixels / diff_x + 1
    scale_y = additional_pixels / diff_y + 1
    result = scale(shapely_polygon, xfact=scale_x, yfact=scale_y)
    return list(result.exterior.coords)
```

Algorithmus 4 Dieser Algorithmus bekommt ein Orthophoto, die von Algorithmus 1 berechneten Werte und einen in Pixel-Koordinaten umgerechneten Umring. Hierbei werden die *NumPy*- (vgl. [81], [82]) und die *Pillow*-Bibliothek (vgl. [84]) verwendet. `scale_polygon` ist die Methode aus Algorithmus 3.

```
1  import numpy as np
2  from PIL import ImageDraw, Image
3
4  def cut_out_polygon(self, x: int, y: int, w: int, h: int, polygon: PxCoordStructure):
5      channels = 3
6      polygon = scale_polygon(polygon)
7      polygon = [(poly_x - x, poly_y - y) for poly_x, poly_y in polygon]
8      cropped = self.cut_out_rect(x, y, w, h)
9      cropped = cropped[:channels, :, :]
10
11     mask = np.zeros((h, w, channels))
12     mode = 'RGB'
13     mask = Image.fromarray(mask, mode=mode)
14     color = tuple([1] * channels)
15     ImageDraw.Draw(mask).polygon(polygon, fill=color)
16     mask = np.moveaxis(np.array(mask), -1, 0)
17     dst = cropped * mask
18     return np.array(dst)
```

multipliziert (Hadamard-Produkt). Dadurch werden Pixel des Dachbildes, die außerhalb des Daches liegen, auf Null gesetzt, und alle anderen erhalten. (vgl. Algorithmus 4)

Dadurch wird das Dachbild genannte Orthophoto manipuliert, die entsprechende PBM-Rastergrafik mit eventuellen Annotationen bleibt aber unverändert.

6 Klassifizierung

Um Solaranlagen in Orthophotos zu lokalisieren, klassifizieren wir jeden Pixel in „enthält Solarmodul“ („positiv“) und „enthält kein Solarmodul“ („negativ“). Das Eingabebild kann beliebige Höhe und Breite haben, das Ausgabebild ist immer entsprechend gleich groß. Weil die von uns verwendeten Eingabebilder georeferenziert sind, können wir auch jedem Pixel des Ausgabebildes Koordinaten zuordnen. Die Auflösung der Klassifizierung entspricht somit der Bodenauflösung der für die Eingabe verwendeten Orthophotos.

Dafür konvertieren wir jeden Pixel zuerst in einen sogenannten *Feature-Vektor* (vgl. Abschnitt 6.1) und ordnen diesem dann mithilfe eines neuronalen Netzes eine Wahrscheinlichkeit zu (vgl. Abschnitt 6.2). Das ist in Abbildung 18 skizziert. Die Wahrscheinlichkeit wird über einen konfigurierbaren Schwellwert zu einer binären Aussage gemacht. Diese werden entsprechend dem ursprünglichen Bild zu einer Rastergrafik mit binären Werten zusammengesetzt.[31] Optional wenden wir danach noch morphologische Operationen an (vgl. Abschnitt 6.3).

In den folgenden Abschnitten werden diese Schritte chronologisch erklärt.

6.1 Berechnung von Feature-Vektoren

Jeden Pixel eines Dachbildes (aufbereitete Eingabedaten, vgl. Abschnitt 5) konvertieren wir in einen *sogenannten* Feature-Vektor, bevor wir diesen weiter verarbeiten. Damit wollen wir das darauf folgende neuronale Netz (vgl. Abschnitt 6.2) auf bestimmte Bildeigenschaften zu trainieren.

Ein Feature-Vektor besteht aus einer Menge von Werten, welche aus den je drei Farbwerten (rot, grün, blau) eines Pixels und der Pixel in dessen Umgebung berechnet werden. Weil nicht nur der Pixel selbst, sondern auch dessen Umgebung miteinbezogen wird, müssen die entsprechenden Pixel bekannt oder extrapolierbar sein. Das setzen wir in dieser Beschreibung voraus. In unserer Implementierung lösen wir das Problem,

[31] Die Rastergrafik mit binären Werten persistieren wir vor der Anwendung von morphologischen Operationen zur Optimierung des Rechenaufwands. Dafür haben wir aufgrund seiner Einfachheit den *Portable-Bitmap*-Standard (PBM, vgl. `http://netpbm.sourceforge.net/doc/pbm.html` (abgerufen am 01.05.2020)) verwendet.

indem wir Bilder verwenden, die größer sind, als der eigentlich betrachtete Bereich.

Wir haben verschiedene Umgebungen und Formeln zum Berechnen der Werte von Feature-Vektoren ausprobiert, welche wir im Folgenden vorstellen.

Umgebung eines Pixels Einen Pixel, der durch die Koordinaten (x, y) gegeben ist, zusammen mit seinen direkt benachbarten Pixeln, bezeichnen wir als Block (3×3 Pixel):

$$\text{Block}_{x,y} = \{(a, b) | a \in \{x-1, x, x+1\} \wedge b \in \{y-1, y, y+1\}\}$$

Eine hier so genannte *Sudoku-Umgebung* um einen Pixel besteht aus mehreren Blöcken, abhängig von einem Radius r:

$$\text{Sudoku}_r(x, y) = \{\text{Block}_{a,b} | a \in \{x-r, x, x+r\} \wedge b \in \{y-r, y, y+r\}\}$$

Ein Beispiel für Blöcke und Sudoku-Umgebungen ist in Abbildung 17 zu sehen. Die verschiedenen Kombinationen aus Sudoku-Umgebungen und optional dem Block, die wir ausprobiert haben, werden in Tabelle 3 aufgelistet.

Werte des Feature-Vektors Für jeden Farbkanal des Eingabebildes wird eine Liste von Werten bestimmt, die konkateniert den Feature-Vektor ergeben. Die Reihenfolge der Werte ist für eine Feature-Vektor-Berechnugsvorschrift immer gleich. Die Farbkanäle werden alle gleich behandelt. Die Werte für einen Farbkanal werden aus der zuvor beschriebenen Umgebung des Pixels berechnet, welche aus mindestens einem Block besteht. Pro Block wählen wir aus diesen Werten aus:

val	die Werte der Pixel im Block
dif	pro Pixel die Differenz vom mittlerem Pixel
avg	das arithmetische Mittel der Pixel des Blocks
var	die Varianz der Pixel des Blocks

Wir haben verschiedene Kombinationen dieser Werte für verschiedene Umgebungen zur Berechnung von Feature-Vektoren ausgewählt. Die Zusammensetzung und Benen-

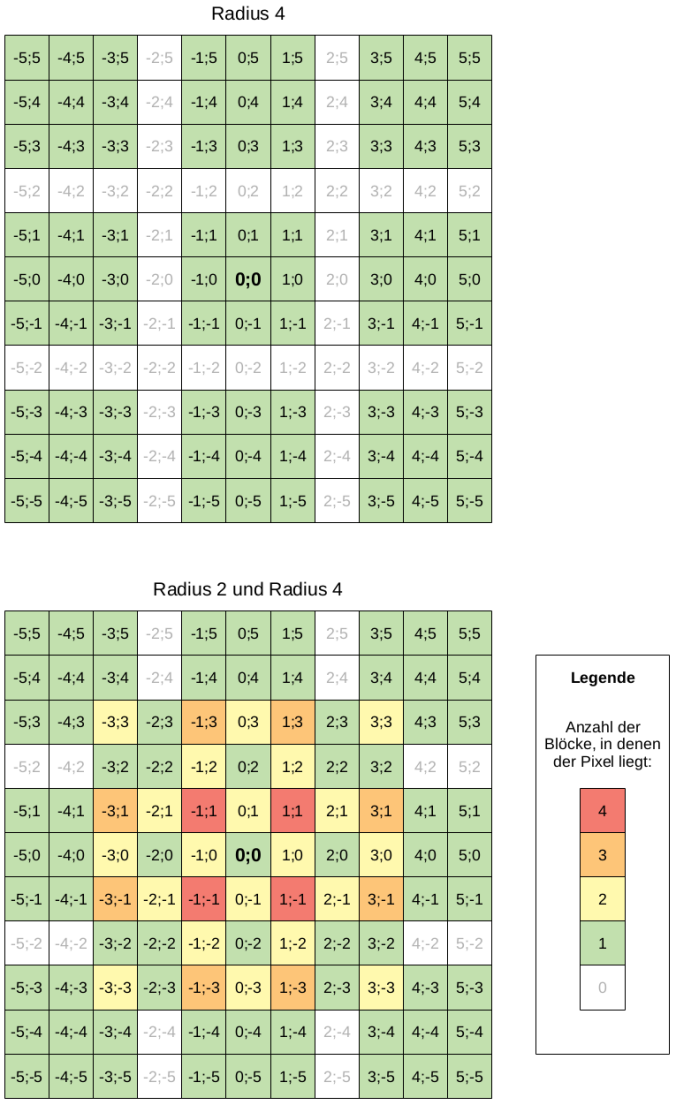

Abbildung 17: Hier sind mehrere *Sudoku-Umgebungen* skizziert, je relativ zum fett markierten Pixel in der Mitte. Die *Blöcke* sind farbig hervorgehoben. Die ausgegrauten Pixel gehören nicht zur Umgebung.
Oben ist eine Sudoku-Umgebung mit Radius 4 zu sehen. Das entspricht dem Feature-Vektor fv4 (vgl. Tabelle 3).
Unten ist eine Sudoku-Umgebung mit Radius 4 und eine Sudoku-Umgebung mit dem Radius 2. Der Block um den zentralen Pixel kommt in beiden Sudoku-Umgebungen vor, wird hier aber nur einfach gewertet dargestellt. Das entspricht dem Feature-Vektor fv24 (vgl. Tabelle 3).

Name	Umgebung pro Farbkanal Block um den Pixel	Sudoku_2	Sudoku_3	Sudoku_4	Anzahl Werte
`fv2`		avg, var			54
`fv3`			avg, var		54
`fv4`				avg, var	54
`fv23`		avg, var	avg, var		102
`fv34`			avg, var	avg, var	102
`fv24`		avg, var		avg, var	102
`fv02`	val, dif	avg, var			108
`fv03`	val, dif		avg, var		108
`fv04`	val, dif			avg, var	108
`fv023`	val, dif	avg, var	avg, var		156
`fv034`	val, dif		avg, var	avg, var	156
`fv024`	val, dif	avg, var		avg, var	156

Tabelle 3: Diese Auswahl an Kombinationen von Werten aus bestimmten Umgebungen haben wir für Feature-Vektoren ausprobiert. Alle Sudoku-Umgebungen beinhalten des gleichen mittleren Block. Wir nehmen diesen jedoch nur einmal in die Feature-Vektoren auf, um keine redundanten Werte einzubringen. Zur Beschreibung der Werte siehe Abschnitt 6.1. Die angegebenen Werte wurden pro Farbkanal berechnet, was bei unseren Eingabebildern 3 (RGB) waren.

nung ist in Tabelle 3 zu finden.

Der Feature-Vektor *fv24* entspricht dem Feature-Vektor, den Malof et al. in ihrer Arbeit verwendet haben (siehe Abschnitt 3.2 und [29]) und hat unsere Berechnung von Feature-Vektoren maßgeblich inspiriert. Die mit *fv24* betrachtete Umgebung ist in Abbildung 17 dargestellt.

6.2 Aufbau des neuronalen Netzes

Das verwendete neuronale Netz (NN) verarbeitet jeden Feature-Vektor (und darüber jeden Pixel) einzeln. Die Werte des Feature-Vektors bilden die Eingabe für das NN. Das besteht aus fünf vollverbundenen Layern, die in Tabelle 4 aufgelistet werden. Der letzte Layer enthält ein einzelnes Neuron, das durch die Aktivierungsfunktion *Sigmoid* auf den Wertebereich $(0, 1)$ abgebildet wird. Dieser Wert ist die Erkennungswahrscheinlich-

Layer	Anzahl Neuronen	Aktivierungsfunktion
L1	200	ReLU
L2	200	ReLU
L3	100	ReLU
L4	10	ReLU
L5	1	Sigmoid

Tabelle 4: Unser neuronales Netz besteht aus diesen vollverbundenen Layern. In den ersten Layer, L1, wird der Feature-Vektor eingegeben. Der Letzte Layer, L5, gibt die Erkennungswahrscheinlichkeit aus. Diese wird danach über einen Schwellwert und ggf. morphologische Operationen gefiltert. Die Implementierung dieser Layer wird in Algorithmus 5 gezeigt.

keit, mit der das NN ein Solarmodul in dem Pixel sieht. Wir haben kein bestehendes NN verwendet, sondern basierend auf Erfahrungen und anderen Arbeiten eine eigene experimentelle Architektur gewählt.

Zum Trainieren des Netzes verwenden wir *Batches* der Größe 16384, *binäre Kreuzentropie* als Loss-Funktion und *Adam* (vgl. [53]) als Optimierungsfunktion. Adam wurde auch schon von Castello et al. (vgl. Abschnitt 3.6) zur Lokalisierung von PV-Modulen verwendet.

Für die Implementation des NN haben wir die *PyTorch*-Bibliothek (vgl. [85]) verwendet, siehe Algorithmus 5.

Zum Trainieren verwenden wir einen 10 000 Dachbilder großen Datensatz. Da wir diesen aufgrund der uns zur Verfügung stehenden Hardware nicht als ganzes verarbeiten konnten, haben wir diesen in 1 000 Dachbilder große Gruppen unterteilt. Jede Epoche haben wir zwei dieser Gruppen verwendet, eine zum Trainieren und eine, um die für die Optimierung nötige Bewertung vorzunehmen. In der darauf folgenden Epoche nehmen wir eine neue Gruppe für die Bewertung und verwenden die zuvor für die Bewertung verwendete Gruppe für das Trainieren. Auf diesem Wege durchlaufen wir den gesamten Datensatz in 10 Epochen.

Algorithmus 5 In dieser Klasse werden die Layer des neuronalen Netzes definiert. Dazu wird die *PyTorch*-Bibliothek (vgl. [85]) verwendet. Der Parameter `input_size` gibt die Anzahl Werte des verwendeten Feature-Vektors an. Die definierten Layer sind ebenso in Tabelle 4 aufgeführt.

```
1  import torch
2  import torch.nn as nn
3  import torch.nn.functional as F
4
5
6  class ImageFeatureNN(nn.Module):
7      def __init__(self, input_size: int):
8          super(ImageFeatureNN, self).__init__()
9          self.fc1 = nn.Linear(input_size, 200)
10         self.fc2 = nn.Linear(200, 200)
11         self.fc3 = nn.Linear(200, 100)
12         self.fc4 = nn.Linear(100, 10)
13         self.fc5 = nn.Linear(10, 1)
14
15     def forward(self, x):
16         x = F.relu(self.fc1(x))
17         x = F.relu(self.fc2(x))
18         x = F.relu(self.fc3(x))
19         x = F.relu(self.fc4(x))
20         x = self.fc5(x)
21         return torch.sigmoid(x)
22
23     def num_flat_features(self, x):
24         size = x.size()[1:]
25         num = 1
26         for i in size:
27             num *= i
28         return num
```

6.3 Morphologische Operationen

Wir haben versucht, falsche Klassifikationen in mit dem zuvor vorgestellten Algorithmus erstellten binären Rastergrafiken mittels morphologischer Operationen zu reduzieren. Dazu haben wir uns die Operationen *Opening* und *Closing* ausgesucht. Diese wurden im gleichen Kontext bereits von Malof et al. und Puttemans et al. eingesetzt (vgl. Abschnitt 3.2.2 und Abschnitt 3.3).

Diese liegen *Dilatationen* und *Erosionen* zugrunde. Beim Opening werden zuerst Erosionen, dann Dilatationen angewendet, und beim Closing umgekehrt. Bei beiden zugrundeliegenden Operationen wird eine Art Schablone („Kernel“) über jeden Pixel der Rastergrafik geschoben. Je nachdem, ob ein bestimmtes Kriterium erfüllt wird, wird der entsprechende Pixel in der Ausgabegrafik eingefärbt (positiv/negativ). Bei der Dilatation ist das Kriterium, dass die Schablone mit *mindestens einem* positiven Pixel überlappt. Bei der Erosion ist das Kriterium, dass die Schablone *nur* mit positiven Pixeln überlappt. (vgl. [18])

Wir haben verschiedene Kombinationen von Opening und Closing ausprobiert,[32] mehr dazu in Abschnitt 7.

[32] Wir haben die Implementierung von *OpenCV* verwendet, vgl. `https://docs.opencv.org/trunk/d9/d61/tutorial_py_morphological_ops.html` (abgerufen am 29.05.2020).

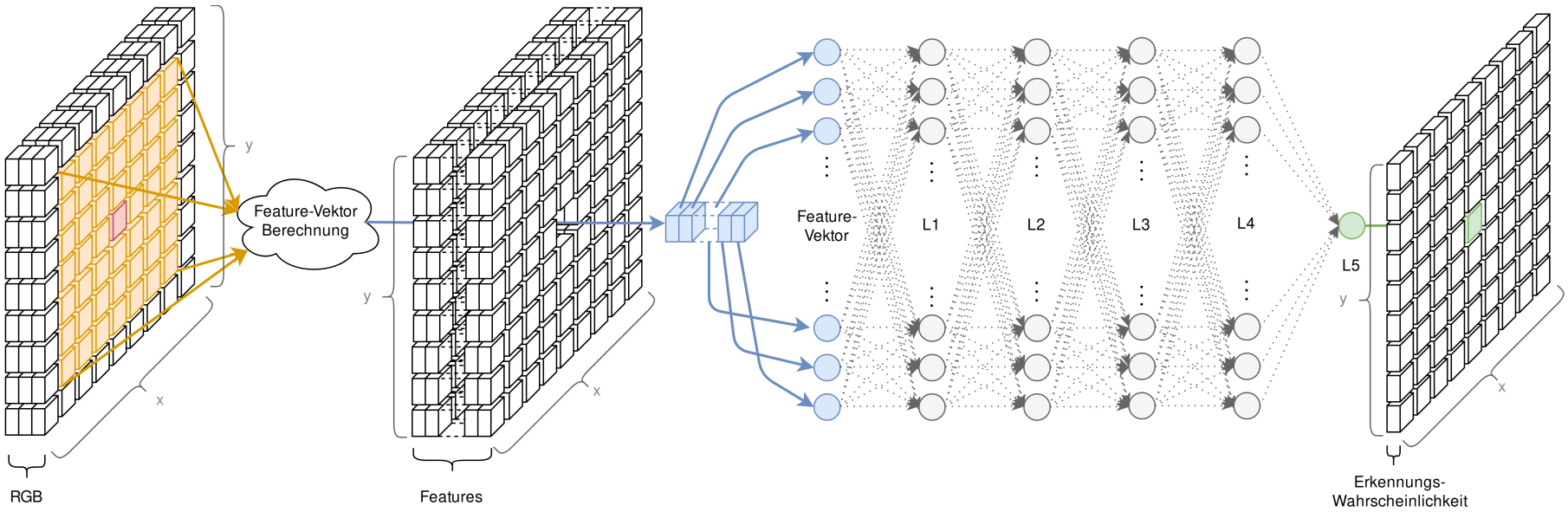

Abbildung 18: Hier ist die Architektur unseres neuronalen Netzes skizziert. Die Eingabe ganz links ist ein Orthophoto mit RGB-Farbkanälen. Daraus wird ein Pixel gewählt (rot), für den dann eine Wahrscheinlichkeit (grün), ob dieser einem PV-Modul entspricht, berechnet wird. Dazu wird zunächst aus dem Pixel, zusammen mit seinen beanchbarten Pixeln (orange), ein Feature-Vektor (blau) berechnet. Die Werte dessen stellen die Eingabe für den ersten Layer des neuronalen Netzes dar. Es gibt fünf Layer, die alle vollverbunden sind. Der Letzte Layer hat nur ein Neuron und somit einen einzelnen Ausgabewert. Jener ist die genannte Wahrscheinlichkeit. Details zu den Layern sind in Tabelle 4 zu finden. Die Maße $x \times y$ des Orthophotos bleiben bei der Menge der Feature-Vektoren und der der Erkennungswahrscheinlichkeiten erhalten.

7 Ergebnisse der Ansätze

Wir haben den in 6 vorgestellten Algorithmus in verschiedenen Variationen mit verschiedenen Datensätzen (vgl. 5) trainiert und evaluiert und stellen die Ergebnisse in diesem Abschnitt vor.

Bei allen Variationen haben wir den Datensatz Kalifornien (vgl. Abschnitt 5.2) zum Trainieren verwendet. Dabei haben wir jedoch variiert, ob die Daten zuvor *geschwärzt* (vgl. Abschnitt 5.3) werden. Wir haben 9 000 zufällige Dachbilder mit mindestens einer positiven Annotation (enthält Solarmodul) zusammen mit 1 000 zufälligen Dachbildern mit nur negativen Annotationen aus dem Datensatz Kalifornien verwendet. Wir haben den Datensatz NRW (vgl. Abschnitt 5.1) nicht zum Trainieren genutzt, weil wir dazu zu wenig Annotationen hatten.

Ein Beispiel für die erfolgreiche Lokalisierung einer PV-Anlage in einem geschwärzten Dachbild ist in Abbildung 19 zu finden.

Die exakten Zahlen zu allen nachfolgend vorgestellten Ergebnissen haben wir auf *GitLab* veröffentlicht, siehe [2].

7.1 Schwärzen der Dachbilder verbessert die Genauigkeit

Als erstes wollten wir herausfinden, ob es sich lohnt, die Dachbilder beim Zuschneiden zu *schwärzen* (vgl. Abschnitt 5.3). Damit soll das neuronale Netz auf die relevanten Bildbereiche fokussiert werden.

Für alle in Betracht gezogenen Feature-Vektoren (vgl. Tabelle 3) haben wir das Netz trainiert und evaluiert, mit und ohne die Dachbilder zu schwärzen. Das Netz haben wir jeweils neu 100 Epochen lang mit dem Datensatz Kalifornien trainiert. Die Evaluation haben wir mit 1 000 anderen, zufälligen Dachbildern aus dem gleichen Datensatz gemacht. Die Hälfte dieser Dachbilder war mit Solarmodulen, die andere ohne. Als Schwellwert (vgl. Abschnitt 6) haben wir 50 % verwendet.

Die Ergebnisse dieses und aller anderen Experimente messen wir anhand der pro Pixel bestimmten korrekten positiven, korrekten negativen, falschen positiven und falschen negativen Klassifikationen. In den Diagrammen stellen wir die daraus berechneten Genauigkeiten (Anteil korrekter an allen positiven Klassifizierungen) über Trefferquoten

(a) Dachbild (b) Annotation

(c) fv04-150ep-co (d) fv24-150ep-co (e) fv024-150ep-co

Abbildung 19: Hier ist beispielhaft ein geschwärztes Dachbild aus dem Datensatz Kalifornien (vgl. Abschnitt 5.2) mit verschiedenen weißen Einzeichnungen zu sehen. Abbildung a zeigt das geschwärzte Dachbild ohne Änderung. In Abbildung b sind zusätzlich die PV-Module entsprechend der vorgegebenen Annotation in weiß eingefärbt. Auf den Abbildungen c, d und e wurden von dem in Abschnitt 6 vorgestellten Algorithmus PV-Module in weiß eingepinselt. Dabei wurden die Feature-Vektoren fv04, fv24 und respektive fv024 verwendet. Bei allen drei wurde das neuronale Netz 150 Epochen lang trainiert, ein Schwellwert von 50 % angewendet und die morphologischen Operationen Closing und dann Opening angewendet. Die Genauigkeiten und Trefferquoten dieser Konfigurationen können Abbildung 22 und Abbildung 23 entnommen werden.

(Anteil erkannter an allen wahren positiven Klassifizierungen) dar. Ziel aller Optimierungen ist, Genauigkeit und Trefferquote gleichzeitig auf 100 % zu bringen, was einer perfekten Klassifizierung entspräche. Je näher ein Punkt in solchen Diagrammen der oberen rechten Ecke ist, desto besser.

Für dieses Experiment sind Genauigkeiten und Trefferquoten in Abbildung 21 zu finden. Die beste mit 72,3 % beste Genauigkeit in diesem Experiment konnten wir bei einer Trefferquote von 47,1 % mit dem Feature-Vektor `fv04` und geschwärzten Dachbildern erzielen. Die beste Trefferquote von 53,7 % erreichte der Feature-Vektor `fv024` ohne geschwärzte Dachbilder, bei einer Genauigkeit von 62,0 %. Den besten Jaccard-Index von 0,42 erreichte der Feature-Vektor `fv024` ohne geschwärzten Hintergrund (Genauigkeit 71,7 %, Trefferquote 50,8 %). Diese Zahlen sind nicht schlecht, bieten aber insbesondere bei der Trefferquote noch deutlich Verbesserungspotential.

Durch das Schwärzen wurde bei allen Feature-Vektoren die Genauigkeit verbessert. Bei der Hälfte der Feature-Vektoren hat sich außerdem die Trefferquote verbessert. Deswegen verwenden wir bei allen folgenden Untersuchungen nur noch geschwärzte Dachbilder (für das Trainieren und Evaluieren).

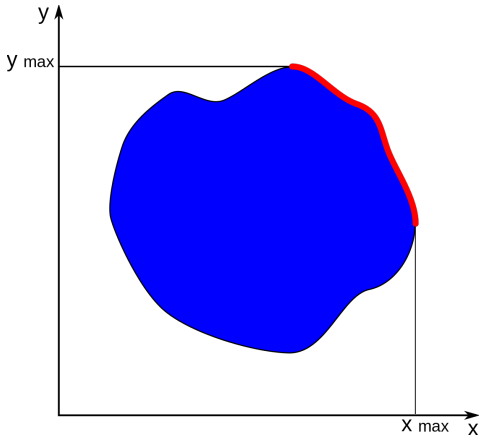

Abbildung 20: Auf der roten Linie sind pareto-optimale Werte aus der blauen Menge. Das ist ein Maß. um bestmögliche Punkte in Abhängigkeit von zwei Variablen zu finden. (vgl. [86], Bildquelle: [87])

7.2 Pareto-optimale Feature-Vektoren

Von der großen Auswahl an Feature-Vektoren (vgl. Tabelle 3) wollten wir herausfinden, mit welchen das neuronale Netz am besten lernt, wo Solarmodule sind. Dafür betrachten wir die Ergebnisse bei geschwärzten Dachbildern aus dem vorherigen Experiment. Daraus zeichnen

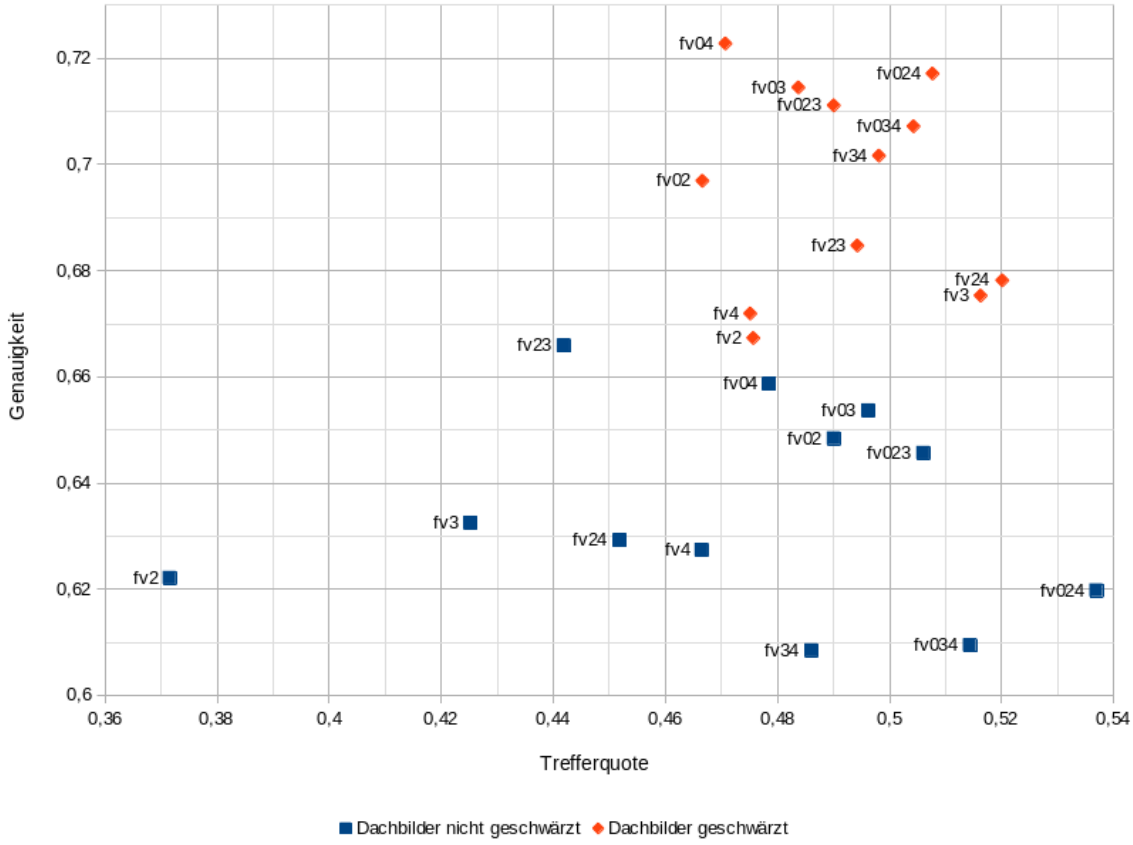

Abbildung 21: Hier sind die Evaluationsergebnisse für verschiedene Feature-Vektor-Berechnungsvorschriften und Eingabedaten dargestellt. Die Namen der Feature-Vektoren sind in Tabelle 3 aufgeschlüsselt. Weitere Informationen zur Berechnung und eine Bewertung sind in Abschnitt 7.1 zu finden.

sich klar drei pareto-optimale (vgl. [86]) Feature-Vektoren ab: `fv04`, `fv024` und `fv24`. Weil das erheblich den Rechenaufwand reduziert, verwenden wir künftig nur noch diese drei Feature-Vektoren.

7.3 Morphologische Operationen und Anzahl Epochen

Als nächstes haben wir verschiedene Kombinationen von morphologischen Operationen (vgl. Abschnitt 6.3) ausprobiert, genauer gesagt: Alle in Tabelle 5 aufgelisteten Kombinationen von morphologischen Operationen. Ziel davon ist, Ausreißer, und damit falsche Klassifikationen, anhand ihrer benachbarten klassifizierten Pixel zu erkennen und zu eliminieren. Dadurch sollen Genauigkeit und Trefferquote verbessert werden.

Wir haben wie zuvor mit geschwärzten Dachbildern aus dem Datensatz Kalifornien trainiert und die drei pareto-optimalen Feature-Vektoren verwendet. Diesmal haben wir das neuronale Netz um mehr Epochen trainiert, und nach 100, 150 und 200 Epochen Evaluationen gemacht.

Die Ergebnisse sind in Abbildung 22 zu sehen. Die Beschriftung der Datenpunkte setzt sich aus dem Feature-Vektor und der Anzahl Epochen zusammen (z.B. entspricht `fv24-150ep` Feature-Vektor `fv24` und 150 Epochen).

Die beste Genauigkeit von 79,8 % erreichte der Feature-Vektor `fv04` bei 200 Epochen Training und *Opening* als morphologische Operation. Dabei erreichte dieser eine Trefferquote von 58,1 %. Die beste Trefferquote von 68,2 % erreichte der Feature-Vektor `fv024` bei 150 Epochen Training und Closing. Diese erreichte er mit einer Genauigkeit von 68,2 %. Den besten Jaccard-Index von 0,54 erreichte der Feature-Vektor `fv024` bei 200 Epochen und Closing, dann Opening (Genauigkeit: 75,2 %, Trefferquote: 65,8 %).

Im Vergleich zu keiner Operation verbessern alle Operationen außer *Closing* die Genauigkeit. *Closing, dann Opening* und *Closing* verbessern die Trefferquote. Allein *Closing, dann Opening* verbessert Genauigkeit und Trefferquote. Bei den anderen Operationen scheint nur das eine zum Preis des anderen verbessert werden zu können.

Um je mehr Epochen man ein neuronales Netz trainiert, desto genauer lernt dieses die Trainingsdaten. Wenn man jedoch zu viele Epochen wählt, besteht die Gefahr der Überanpassung. Dann erkennt das erkennt das Netz nur noch die Trainingsdaten und

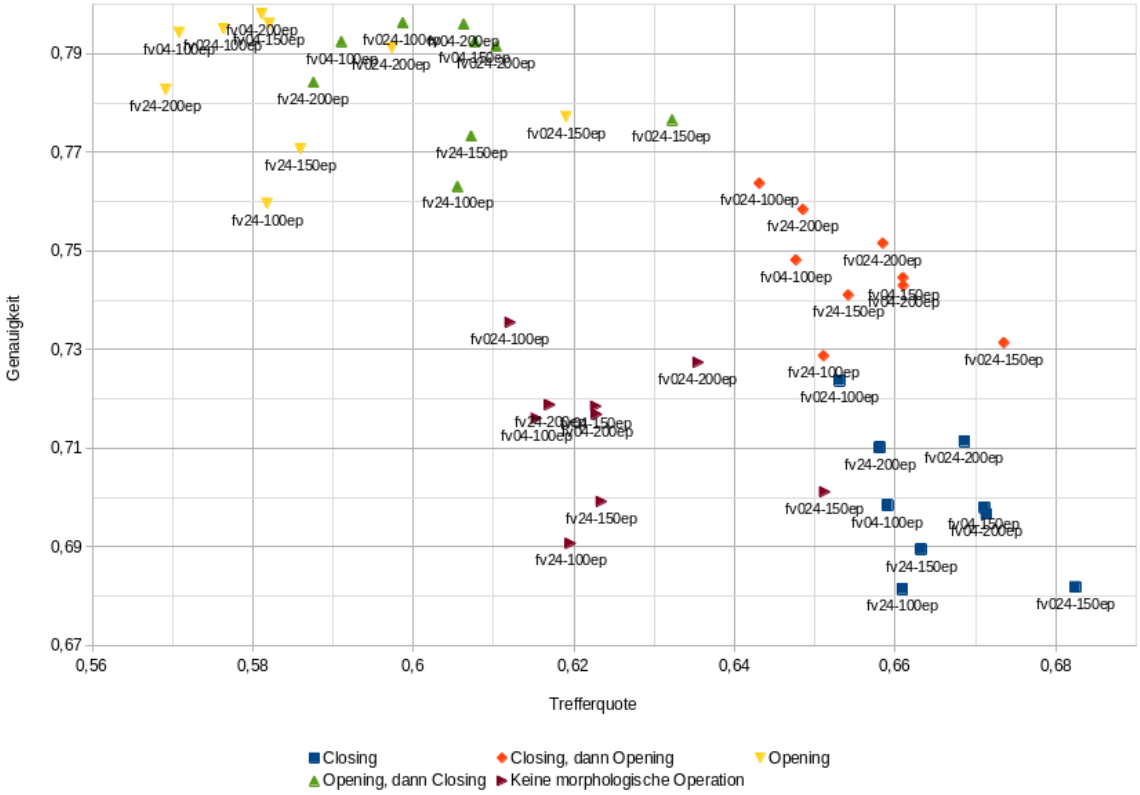

Abbildung 22: Evaluation verschiedener Feature-Vektoren und morphologischer Operationen. Die identischen Werte sind in Abbildung 23 nach Anzahl Epochen kategorisiert dargestellt.

kann nicht zu den Evaluationsdaten hin abstrahieren. Darum ist ein geeigneter Kompromiss zwischen zu viel und zu wenig Epochen benötigt. Wir haben mit 100 Epochen angefangen und dann zwei Mal um 50 Epochen erhöht.

Die Auswertung der verschiedenen Anzahlen Epochen haben wir zugleich mit der Evaluation von morphologischen Operationen ausgeführt. Deswegen sind in Abbildung 22 bereits die Ergebnisse für verschiedene Epochen enthalten. Zur einfacheren Auswertung haben wir selbige Ergebnisse erneut, diesmal nach Anzahl Epochen aufgeschlüsselt, in Abbildung 23 dargestellt.

`c`	*Closing*
`co`	*Closing*, dann *Opening*
`o`	*Opening*
`oc`	*Opening*, dann *Closing*
`x`	keine morphologische Operation

Tabelle 5: Kürzel für morphologische Operationen (vgl. Abschnitt 7.3 und Abschnitt 6.3)

Es ist sofort zu erkennen, dass durch mehr Epochen keine große Verbesserung von Genauigkeit oder Trefferquote einhergeht.

Von 100 zu 150 Epochen verbessert sich die Trefferquote um höchstens 4,3 % (bei `fv024` mit Opening). Die größte Verbesserung der Genauigkeit von 1,2 % erreicht der Feature-Vektor `fv24` mit Closing und dann Opening. Sieben der 15 evaluierten Kombinationen von Feature-Vektor und morphologischen Operationen haben sich in Puncto Genauigkeit verschlechtert. (vgl. Abbildung 24)

Von 150 zu 200 Epochen sieht es nicht besser aus: Keine Trefferquote konnte verbessert werden. Der Feature-Vektor fv024 mit Closing konnte 3,0 % die größte Verbesserung der Genauigkeit erzielen. Drei der Kombinationen haben sich in der Genauigkeit verschlechtert. (vgl. Abbildung 25)

Aufgrund dieser Zahlen haben wir uns dazu entschlossen, für die nachfolgenden Berechnungen mit 150 Epochen trainierte neuronale Netze zu nehmen.

7.4 Auswirkungen von Schwellwerten

Wie in Abschnitt 6 beschrieben, wird die Erkennungswahrscheinlichkeit (Wert zwischen 0 und 1), die das neuronale Netz ausgibt, über einen Schwellwert in eine binäre Klas-

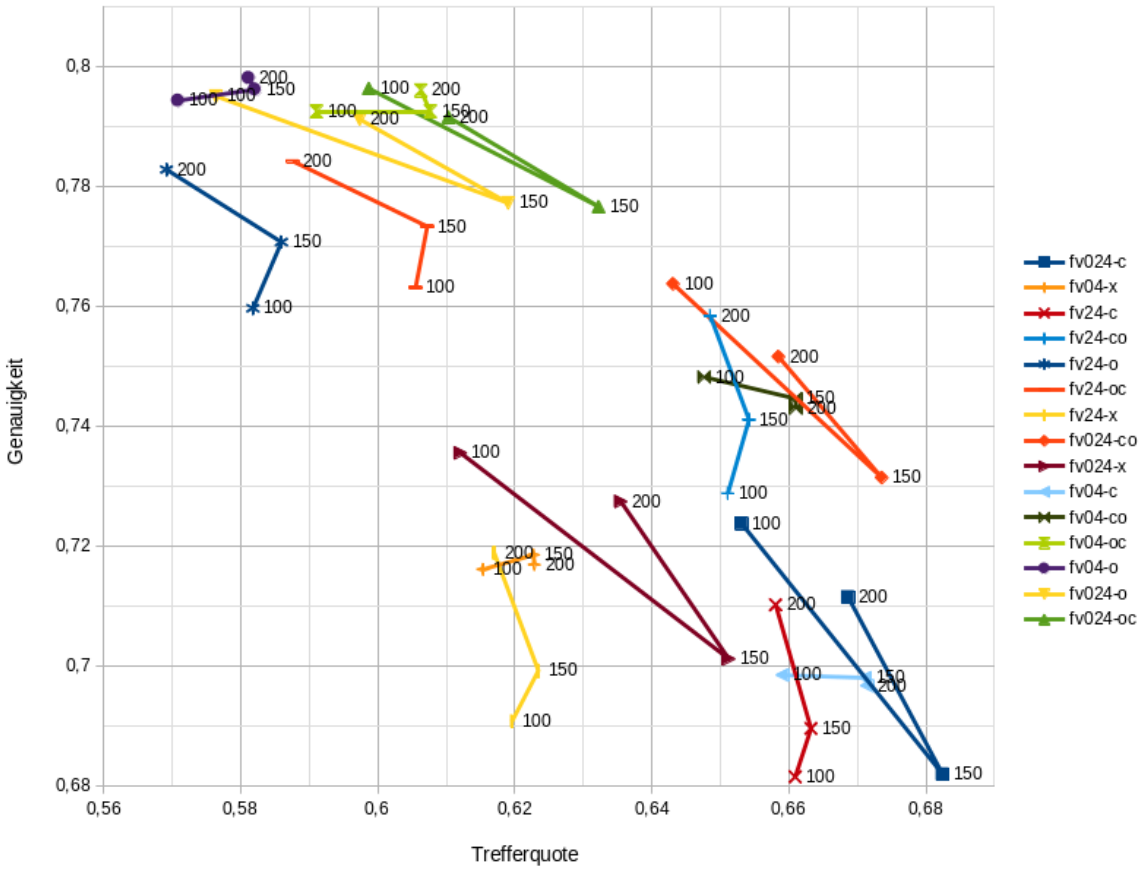

Abbildung 23: Evaluation verschiedener Anzahlen Epochen (Datenpunkt-Beschriftung), für die das neuronale Netz trainiert wurde. Jede Linie entspricht einer Kombination von Feature-Vektor (vgl. Tabelle 3) und morphologischen Operationen (vgl. Tabelle 5). Zur Beschreibung und Auswertung siehe Abschnitt 7.3. Die identischen Werte sind in Abbildung 22 nach morphologischer Operation kategorisiert dargestellt.

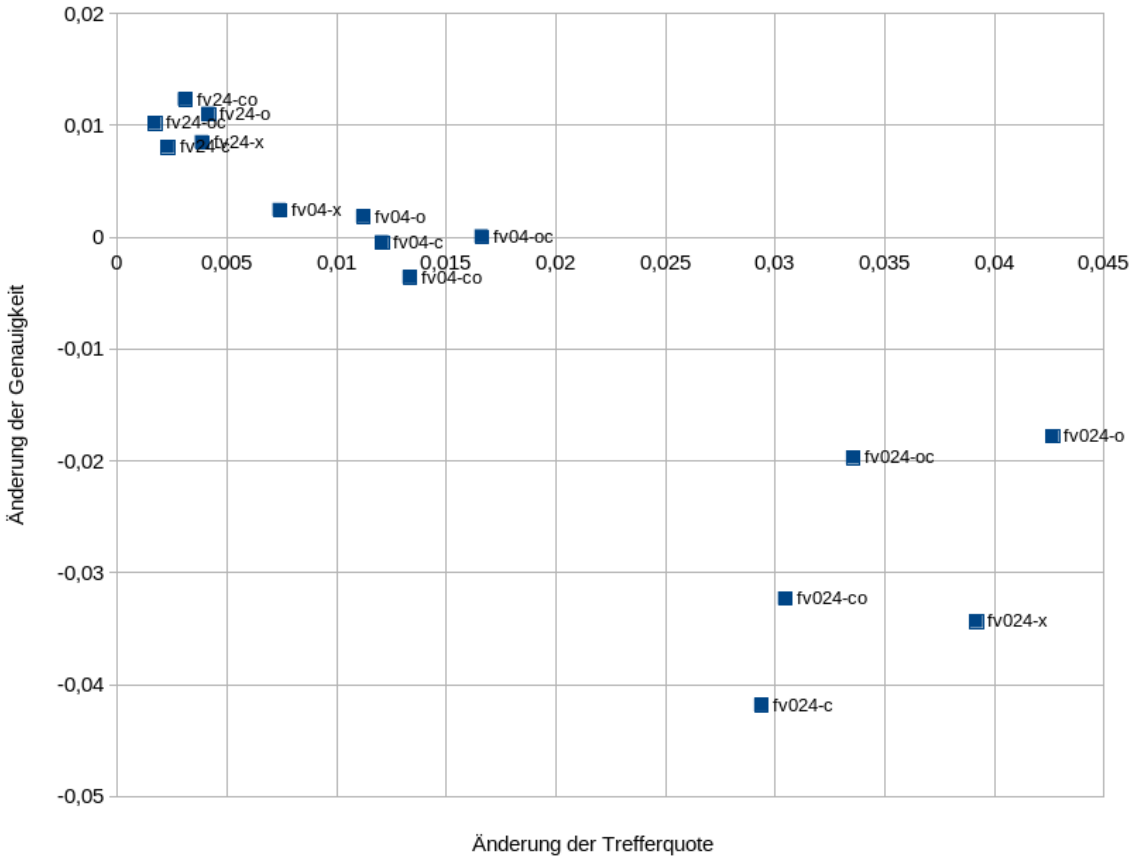

Abbildung 24: Änderung der Evaluations-Ergebnisse von 100 zu 150 Epochen. Zur Auswertung siehe Abschnitt 7.3. Die Beschriftung der Datenpunkte setzt sich aus Feature-Vektor (vgl. Tabelle 3) und morphologischen Operationen (vgl. Tabelle 5) zusammen.

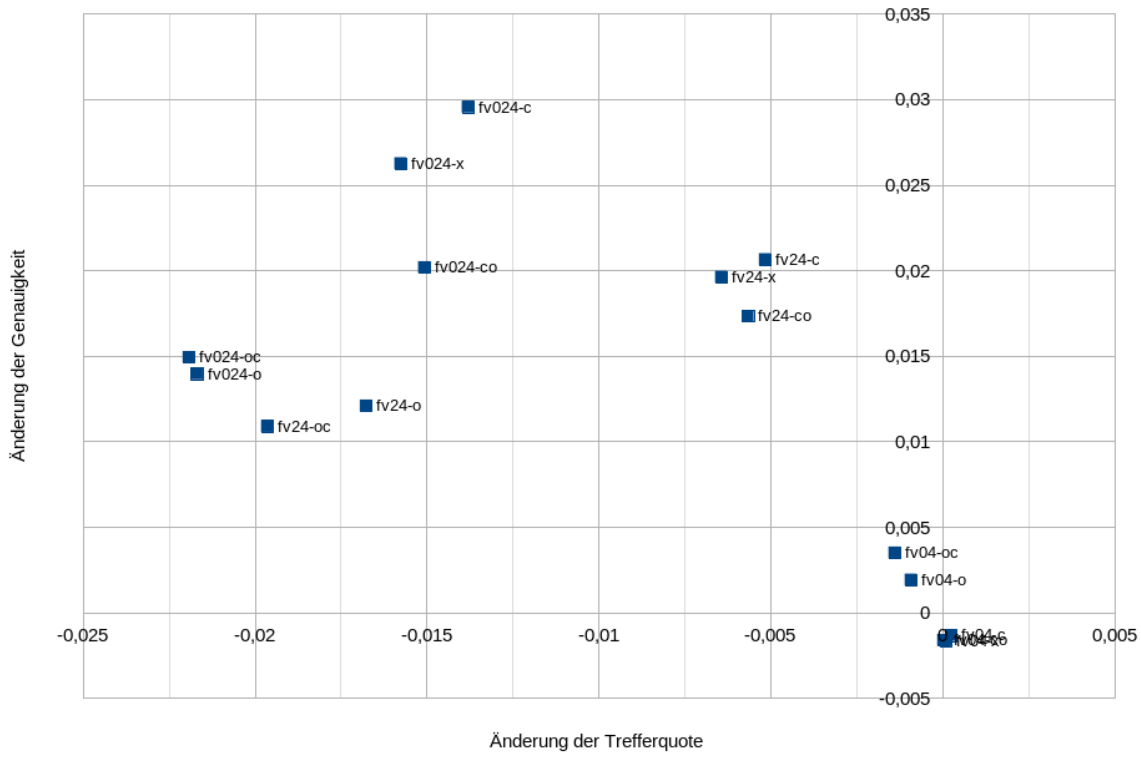

Abbildung 25: Änderung der Evaluations-Ergebnisse von 150 zu 200 Epochen. Zur Auswertung siehe Abschnitt 7.3. Die Beschriftung der Datenpunkte setzt sich aus Feature-Vektor (vgl. Tabelle 3) und morphologischen Operationen (vgl. Tabelle 5) zusammen.

sifikation umgewandelt. Der Schwellwert hat somit maßgeblichen Einfluss auf die Trefferquote. Senkt man den Schwellwert, ist eine Erhöhung der Trefferquote zu erwarten. Allerdings bekommt man diese meist auf Kosten der Genauigkeit. Um einen geeigneten Mittelweg für den Schwellwert zu finden, haben wir verschiedene Schwellwerte evaluiert.

Wir haben in 10 %-Schritten von 10 % bis 80 % alle möglichen Schwellwerte ausprobiert. Deren Evaluation haben wir mit den drei pareto-optimalen Feature-Vektoren und allen in Abschnitt 6.3 genannten morphologischen Operationen durchgeführt. Dabei haben wir das 150 Epochen lang mit Dachbildern aus dem Datensatz Kalifornien trainierte neuronale Netz verwendet.

Den besten Jaccard-Index von 0,54 erreichte der Feature-Vektor `fv04` bei einem Schwellwert von 20 % und Opening, dann Closing. Diese Kombination erreichte eine Genauigkeit von 72,2 % bei einer Trefferquote von 68,9 %.

An den Ergebnissen (siehe Abbildung 26) ist gut zu erkennen, dass Genauigkeit und Trefferquote nicht gleichzeitig über den Schwellwert optimiert werden können. Die beste Genauigkeit, aber auch die schlechteste Trefferquote jeder Kurve wird über den größten Schwellwert erreicht. Umgekehrt wird die beste Trefferquote bei schlechtester Genauigkeit über den geringsten Schwellwert erreicht.

7.5 Evaluation mit dem Datensatz NRW

Bisher haben wir alle Evaluationen mit 1000 Dachbildern aus dem Datensatz Kalifornien (vgl. Abschnitt 5.2) gemacht. Um die Anwendbarkeit unserer Algorithmen auf Datensätze zu deutschen Gegenden zu prüfen, haben wir auch den Datensatz NRW (vgl. Abschnitt 5.1) zur Evaluation herangezogen. Dieser umfasst 143 geschwärzte Dachbilder mit Solarmodulen (positiv) und 1638 ohne (negativ). Die folgenden Evaluationen haben wir stets für alle im vorherigen Abschnitt beschriebenen Kombinationen von Feature-Vektor, morphologischen Operationen und Schwellwerten durchgeführt.

Als erstes haben wir versucht, die Evaluation mit allen Dachbildern des Datensatzes NRW anzustellen. Im Unterschied zum bisher für Evaluationen verwendeten Datensatz haben wir damit deutlich weniger positive Daten und deutlich weniger positive Daten im Verhältnis zu den negativen Daten.

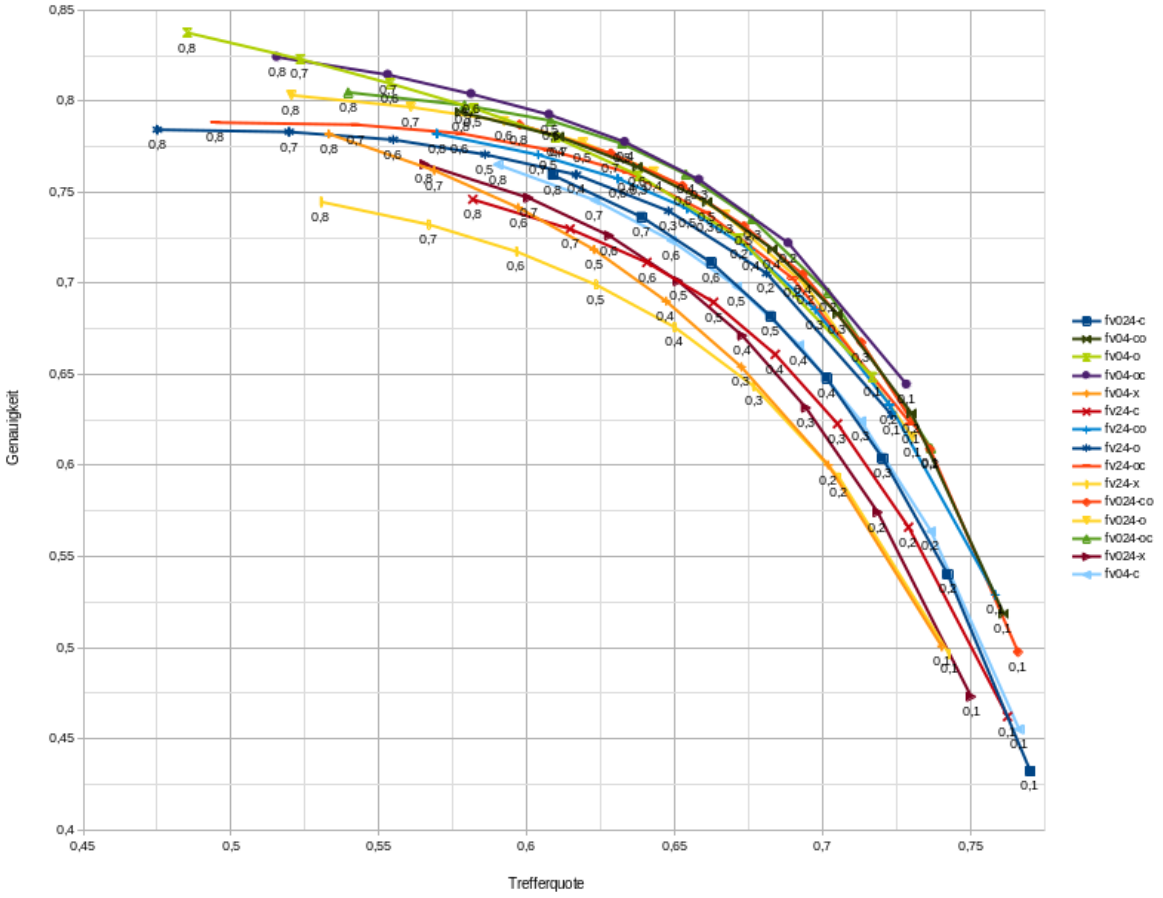

Abbildung 26: Evaluation von verschiedenen Schwellwerten. Zur Auswertung siehe Abschnitt 7.4.
Für jede Kombination aus Feature-Vektor (vgl. Tabelle 3) und morphologischen Operationen (vgl. Tabelle 5) ist eine Linie gezeichnet. Die Beschriftungen an den Datenpunkten sind die Schwellwerte.

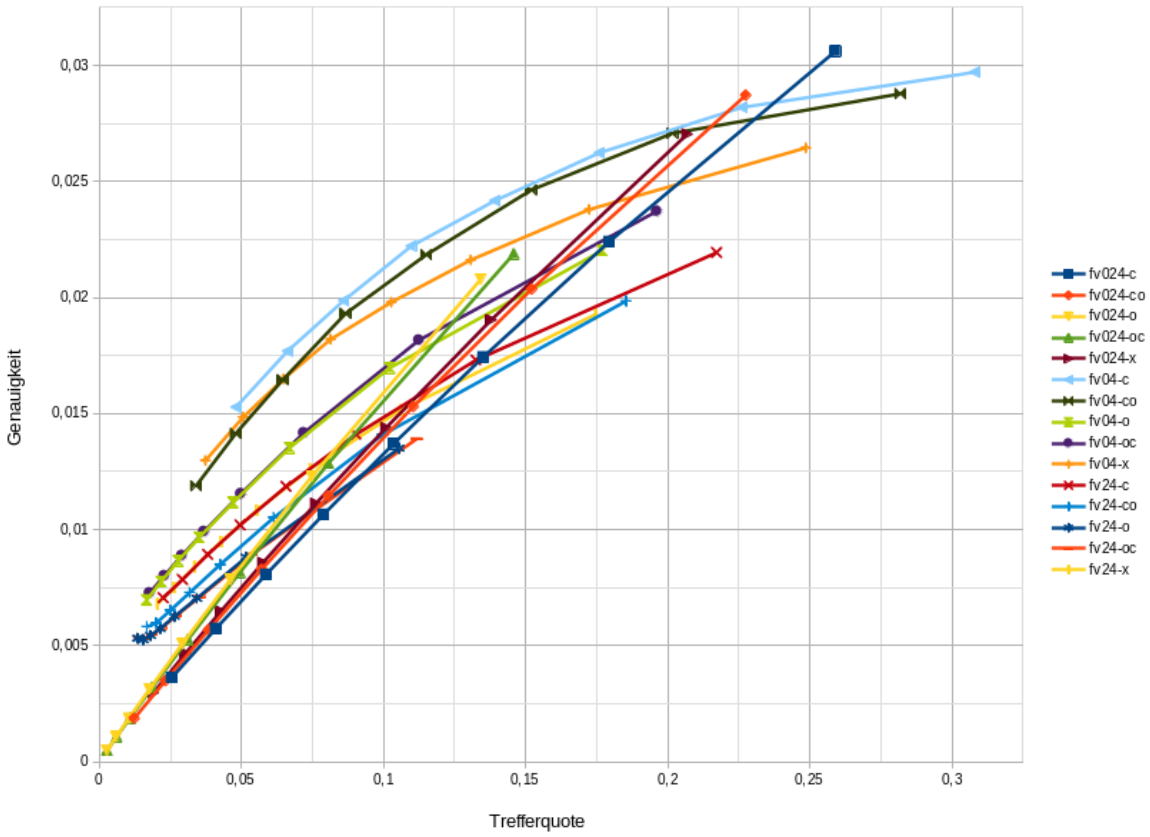

Abbildung 27: Durch Evaluieren mit allen, unveränderten Dachbildern des Datensatzes NRW (vgl. Abschnitt 5.1) konnten wir nur schlechte Ergebnisse erzielen. Zur Auswertung siehe Abschnitt 7.5.
Für jede Kombination aus Feature-Vektor (vgl. Tabelle 3) und morphologischen Operationen (vgl. Tabelle 5) ist eine Linie gezeichnet. Die Datenpunkte davon wurden über verschiedene Schwellwerte erzielt.

Damit konnten wir nur sehr schlechte Ergebnisse erreichen. Die beste Genauigkeit war 3,1 % bei einer Trefferquote von 25,9 % und wurde von Feature-Vektor `fv024` mit Closing und einem Schwellwert von 10 % erzielt. In dieser Konfiguration wurde auch der beste Jaccard-Index von 0,028 erlangt. Die beste Trefferquote von 30,8 % bei einer Genauigkeit von 3,0 % wurde von Feature-Vektor `fv04` mit Closing und einem Schwellwert von 10 % erzielt. Wir haben drei Ideen, woran das liegen könnte:

- Der zum Trainieren verwendete Datensatz (Kalifornien) basiert auf Orthophotos mit einer Bodenauflösung von 30 cm, der zum Evaluieren verwendete Datensatz (NRW) basiert hingegen auf 10 cm Bodenauflösung. Über die Feature-Vektoren werden bei der Klassifizierung eines Pixel dessen benachbarte Pixel mit einbezogen. In welcher Entfernung sich die benachbarten Pixel gemessen in Metern entlang des Bodens befinden, hängt von der Bodenauflösung ab. Die größte Entfernung eines betrachteten Pixels ist bei unseren Feature-Vektoren 5 Pixel. Das entspricht 1,5 m bei 30 cm Bodenauflösung (Trainingsdaten), aber nur 0,5 m bei 10 cm Bodenauflösung (Evaluationsdaten). Das neuronale Netz hat also die Farbwerte von Pixeln in einer bestimmten Entfernung gelernt, wird nun aber mit den Farbwerten von Pixeln in einer ganz anderen Entfernung konfrontiert.

- Das hier verwendete Verhältnis von 143:1638 von positiven zu negativen Evaluationsdaten weicht deutlich vom bisher verwendeten Verhältnis von 500:500 ab. Der Unterschied ist noch stärker im Vergleich mit dem Verhältnis 9000:1000 der Trainingsdaten. Das neuronale Netz hat im Verhältnis viel weniger negative Beispiele gelernt, als nun evaluiert wurden. Weil dadurch bei der Evaluation besonders viele falsche positive Klassifizierungen auftreten, ist dieses Verhältnis keine faire Bewertungsmethode.

- Wie in Abschnitt 2.1 und Abschnitt 4.2 angedeutet, hängt die Vergleichbarkeit von Orthophotos von verschiedensten Faktoren ab, wie z.B. den meteorologischen Bedingungen während der Aufnahme, dem Zeitpunkt der Aufnahme und der Qualität der perspektivischen Entzerrung. Den Datensätzen Kalifornien und NRW liegen völlig verschiedene Orthophotos zugrunde. Da nur der eine Datensatz zum

Trainieren verwendet wurde, hatte das neuronale Netz keine Gelegenheit, das Aussehen des anderen Datensatzes zu lernen, was sich schlecht in den Evaluationsergebnissen damit niederschlägt.

Deswegen haben wir die Anzahl Dachbilder ohne Solarmodule in den Evaluationsdaten auf die Anzahl der Dachbilder mit Solarmodule (143) herabgesetzt. Und wir haben diese Dachbilder einschließlich der zugehörigen PBM-Rastergrafiken mit den Annotationen vor der neuen Evaluation auf 33 % ihrer Größe herab skaliert.[33] Damit haben wir die Bodenauflösung des Datensatzes NRW auf die Bodenauflösung des Datensatzes Kalifornien angeglichen. Die beiden Maßnahmen haben wir separat und zusammen ausprobiert.

Nur das Verhältnis der Evaluationsdaten anzupassen, hat die beste Trefferquote (30,8 %) nicht verbessert. Nur oder zusätzlich die Dachbilder zu skalieren hat die beste Trefferquote deutlich verbessert, in beiden Fällen auf 50,0 %. In allen Fällen wurde die Beste Trefferquote mit dem Feature-Vektor `fv04`, Closing und 10 % als Schwellwert erreicht.

Die beste Genauigkeit (3,1 %) konnte in allen Fällen verbessert werden. Nur die Dachbilder zu skalieren, hat die beste Genauigkeit auf 7,6 % bei einer Trefferquote von 10,9 % angehoben. Das war bei Feature-Vektor `fv024`, Closing und einem Schwellwert von 80 %. Nur das Verhältnis der Evaluationsdaten anzupassen hat die beste Genauigkeit auf 15,0 % erhöht, zugleich mit der dabei besten Trefferquote von 30,8 %. Dafür war der Feature-Vektor `fv04` mit Closing und 10 % Schwellwert verantwortlich. Mit beiden Maßnahmen gleichzeitig konnten wir die insgesamt (für den Datensatz NRW) beste Genauigkeit von 36,9 % bei einer Trefferquote von 10,9 % erzielen. Das wurde mit dem Feature-Vektor `fv024`, Opening und dann Closing und einem Schwellwert von 80 % erreicht.

Der beste Jaccard-Index konnte wie die beste Genauigkeit in allen Fällen verbessert werden. Bei nur skalierten Dachbildern ergab der Feature-Vektor `fv024` mit 80 % Schwellwert, Closing und dann Opening den besten Jaccard-Index von 0,048. Wurde nur das Verhältnis der Dachbilder angepasst, wurde der beste Jaccard-Index von 0,112 mit

[33] Sowohl die Orthophotos (in diesem Fall im JPEG-Format) als auch die PBM-Rastergrafiken haben wir mit dem Programm *imagemagick* (vgl. [88]) skaliert.

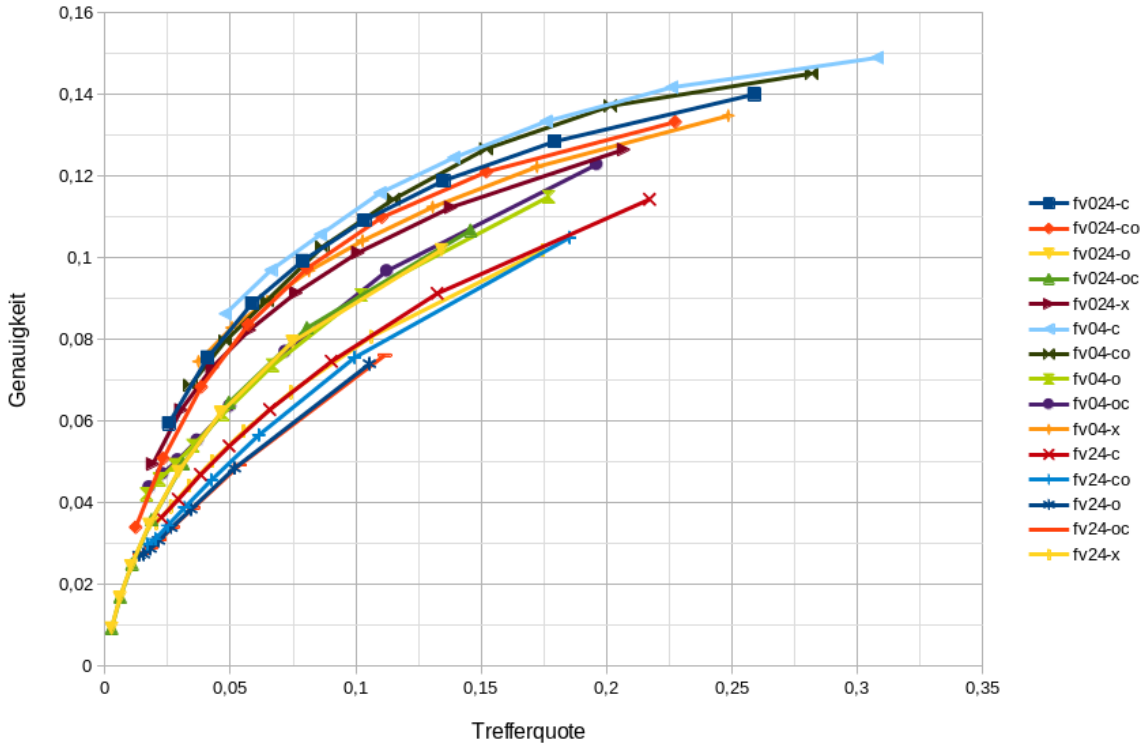

Abbildung 28: Evaluation mit auf 143:143 *angepassten Verhältnis* von positiven und negativen Dachbildern des Datensatzes NRW (vgl. Abschnitt 5.1). Hierfür wurden die Dachbilder nicht skaliert. Zur Auswertung siehe Abschnitt 7.5.
Für jede Kombination aus Feature-Vektor (vgl. Tabelle 3) und morphologischen Operationen (vgl. Tabelle 5) ist eine Linie gezeichnet. Die Datenpunkte davon wurden über verschiedene Schwellwerte erzielt.

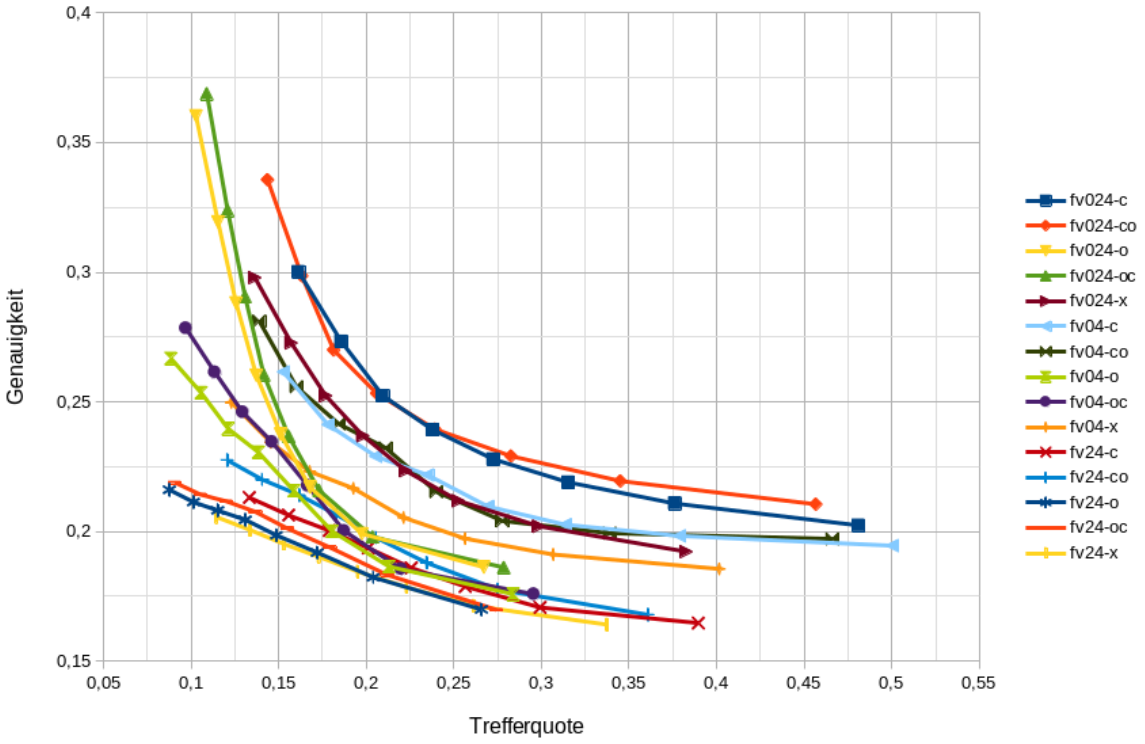

Abbildung 29: Evaluation mit auf 143:143 *angepassten Verhältnis* von positiven und negativen Dachbildern des Datensatzes NRW (vgl. Abschnitt 5.1), die *auf ein Drittel ihrer Größe skaliert* wurden. Zur Auswertung siehe Abschnitt 7.5.
Für jede Kombination aus Feature-Vektor (vgl. Tabelle 3) und morphologischen Operationen (vgl. Tabelle 5) ist eine Linie gezeichnet. Die Datenpunkte davon wurden über verschiedene Schwellwerte erzielt.

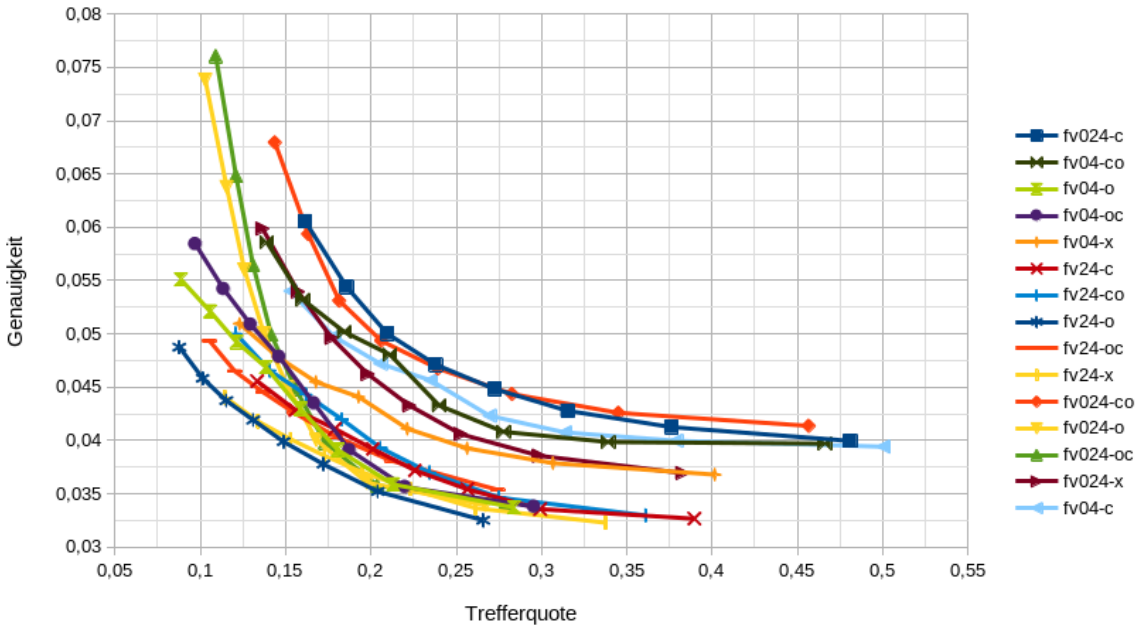

Abbildung 30: Evaluation *mit allen* Dachbildern des Datensatzes NRW (vgl. Abschnitt 5.1), die *auf ein Drittel ihrer Größe skaliert* wurden. Zur Auswertung siehe Abschnitt 7.5.
Für jede Kombination aus Feature-Vektor (vgl. Tabelle 3) und morphologischen Operationen (vgl. Tabelle 5) ist eine Linie gezeichnet. Die Datenpunkte davon wurden über verschiedene Schwellwerte erzielt.

der gleichen Konfiguration wie bei der besten Genauigkeit erzielt. Beide Maßnahmen zusammen führten zu einem besten Jaccard-Index von 0,168 durch den Feature-Vektor `fv024`, 10 % Schwellwert und Closing, dann Opening.

Alle mit dem Datensatz NRW erzielten Evaluationsergebnisse liegen deutlichen hinter den mit dem Datensatz Kalifornien erreichten Evaluationsergebnissen zurück. Das zeigt, dass der gewählte Algorithmus, *trainiert* mit den kalifornischen Daten nicht für die Anwendung auf den Datensatz NRW geeignet ist. Unter Umständen könnte der Algorithmus bessere Ergebnisse erzielen, wenn er ebenfalls mit dem Datensatz NRW *trainiert* würde, da Trainieren und Evaluieren mit dem Datensatz Kalifornien gut funktioniert hat. Wir hatten davon abgesehen, den Algorithmus mit dem Datensatz NRW zu trainieren, weil dieser zu klein war.

8 Fazit

Die Bundesrepublik Deutschland hat sich die Energiewende zum Ziel gesetzt. Infolge dessen hat die Stromversorgung mittels Photovoltaik (PV) einen nennenswerten Anteil und nimmt zu. (vgl. [3])

Daraus ergeben sich verschiedene Gründe, warum die genauen Standorte von PV-Anlagen interessant sind. Einer davon ist es, anhand der Standorte präzise Schätzungen zur momentanen und zukünftigen mit PV erzeugten Leistung anzustellen, um die politischen Ziele der Versorgungssicherheit und Wirtschaftlichkeit sicherzustellen (vgl. [4]). Ein anderer Grund wäre, herauszufinden, welche Umstände zum Kauf von PV-Anlagen motivieren (vgl. [19]).

Im Marktstammdatenregister werden PV-Anlagen mit Koordinaten und Orientierung erfasst, allerdings aus datenschutzrechtlichen Gründen nicht (im benötigten Detailgrad) veröffentlicht. Deswegen verwenden wir Orthophotos (georeferenzierte Luftbilder) und Kartendaten zur Lokalisierung von PV-Anlagen.

Aufgrund des ähnlichen Aussehens differenzieren wir nicht zwischen PV-Modulen und Solarkollektoren, obwohl letztere nicht (direkt) zur Stromerzeugung verwendet werden.

Die Auflösung unserer Lokalisierung entspricht der Auflösung der verwendeten Orthophotos, in unserem Fall 30 cm.

Aus den Kartendaten verwenden wir georeferenzierte Gebäudegrundrisse, um den zu durchsuchenden Bereich auf Gebäude einschränken. Wir suchen nur nach Solaranlagen, die auf Gebäudedächern installiert sind.

Es gibt bereits verschiedene wissenschaftliche Arbeiten, die sich mit einer ähnlichen Zielsetzung wie wir beschäftigt haben. Wir konnten vier Forschergruppen finden, versucht haben, Solaranlagen anhand von Orthophotos in den USA zu lokalisieren, und je eine Forschergruppe die selbiges für Belgien und die Schweiz versucht haben. Zu Deutschland konnten wir keine entsprechende Arbeit finden. Dabei wurde stets maschinelles Lernen eingesetzt, häufig neuronale Netze. Die erzielten Qualitäten der Lokalisierung liegen im praktisch verwendbaren Bereich oder ebnen den Weg dahin. Es wurden viele verschiedene Algorithmen und Bewertungsmethoden ausprobiert, was es

allerdings auch schwierig macht, die Arbeiten zu vergleichen. Für eine detailliertere Zusammenfassung siehe Abschnitt 3.7.

8.1 Datenquellen

Um Solaranlagen in Orthophotos zu lokalisieren, benötigen wir offensichtlich Orthophotos. In Abschnitt 4.2 haben wir eine Recherche zur Verfügbarkeit und Qualität von Orthophotos von Deutschland angestellt. Die Bundesländern erstellen solche Datensätze, allerdings in diverser Qualität, Aktualität und unter verschiedensten Nutzungsbedingungen. Aufgrund der freien Verfügbarkeit und guten Bodenauflösung haben wir uns dazu entschieden, Orthophotos von NRW zu verwenden (vgl. Abschnitt 4.2.1). Dazu haben wir manuell Annotationen, wo sich Solaranlagen befinden, erstellt. Diese haben wir auf *GitLab* veröffentlicht, siehe [2].

Für das von uns verwendete neuronale Netz (NN) benötigen wir nicht nur Orthophotos, sondern auch pixelgenaue Annotationen, wo sich darin Solarmodule befinden. Damit wird das NN trainiert und geprüft. So einen Datensatz zu erstellen, kostet viel händische Arbeit. Deswegen haben wir nur einen kleinen Datensatz zum Prüfen des NN erstellt.

Zum Trainieren und größtenteils zum Prüfen des NN haben wir uns an einem bereits existierenden Datensatz bedient (vgl. Abschnitt 4.2.1 und [1]). Dieser enthält keine Orthophotos von Deutschland, sondern von Kalifornien, USA.

Für unseren Lokalisierungs-Algorithmus benötigen wir außerdem georeferenzierte Umringpolygone von Gebäudegrundrissen. Diese beziehen wir aus *OpenStreetMap* (für Deutschland) und einem von *Microsoft* (für Kalifornien, USA) veröffentlichten Datensatz (vgl. Abschnitt 4.3.1).

8.2 Algorithmus zur Lokalisierung von Solaranlagen

Mit unserem Algorithmus haben wir uns maßgeblich an den Veröffentlichungen von Malof et al. (vgl. Abschnitt 3.2) orientiert. Von ihnen haben wir zum Beispiel die Berechnung von sogenannten Feature-Vektoren adaptiert und evaluiert (vgl. Abschnitt 6.1).

Unser Algorithmus beginnt damit, die zu durchsuchenden Daten zu reduzieren. Wir schneiden die verwendeten Orthophotos so zu, dass je kaum mehr als ein Gebäudedach enthalten ist. Die Positionen dieser erhalten wir aus den bereits erwähnten Kartendaten. Da sich Gebäude für gewöhnlich nicht passgenau in rechteckigen Rastergrafiken abbilden lassen, sind noch Pixel in den Daten, die nicht einem Gebäudedach entsprechen. Diese färben wir schwarz ein. Wir haben herausgefunden, dass diese „Schwarzmalerei“ die Ergebnisse deutlich verbessert (vgl. Abschnitt 7.1).

Jeden Pixel der aufbereiteten Eingabedaten konvertieren wir in einen sogenannten Feature-Vektor. Dabei werden nicht nur der Pixel selbst, sondern auch dessen benachbarte Pixel und somit Kontext einbezogen. Wir haben 12 Berechnungsvorschriften für Feature-Vektoren evaluiert, von denen wir drei pareto-optimale benennen können (vgl. Abschnitt 7.2).

Je ein Feature-Vektor wird dann in ein neuronales Netz gegeben, welches dazu eine Wahrscheinlichkeit ausgibt, ob es sich dabei um ein Solarmodul handelt. Dieser Teil ist in in Abbildung 18 auf Seite 55 illustriert. Bei 150 Epochen Training des neuronalen Netzes haben wir gute Ergebnisse erzielt, mehr Epochen waren kontraproduktiv. Über einen konfigurierbaren Schwellwert wird die ausgegebene Wahrscheinlichkeit zu einer binären Aussage gemacht.

Abschließend, um Ausreißer zu eliminieren, wenden wir noch morphologische Operationen auf den binären Aussagen an. Unsere Evaluation hat ergeben, dass Closing und dann Opening geeignet ist, um die Ausgabequalität zu verbessern (vgl. auf Seite 60).

8.3 Erzielte Qualität der Lokalisierung von Solaranlagen

Unseren Algorithmus haben wir mit Orthophotos mit einer Bodenauflösung von 30 cm trainiert. Daher rühren die 30 cm Bodenauflösung unserer Lokalisierung. Die Ergebnisse unseres Algorithmus bewerten wir auf den Pixel genau, also mit der genannten Bodenauflösung.

Bei der Evaluation mit einem Datensatz, der aus der gleichen Quelle wie die Trainingsdaten stammt, kommen wir auf eine Genauigkeit von 72,2 % bei einer Trefferquote von 68,9 % (vgl. Abschnitt 7.4). Dieses Wertepaar haben wir aus einer Testreihe ausge-

wählt, weil es davon den besten Jaccard-Index von 0,54 hat. Diese Zahlen zeigen, dass der gewählte Algorithmus vielversprechend ist, aber noch deutliches Verbesserungspotential birgt.

Durch die Evaluation mit dem eigens zu NRW erstellten Datensatz konnten wir nur eine Genauigkeit von 21,0 % bei einer Trefferquote von 45,7 % erlangen (vgl. Abschnitt 7.5). Auch dieses Wertepaar haben wir aufgrund des besten Jaccard-Indizes von 0,17 genannt. Diese Ergebnisse sind deutlich schlechter als die anderen und legen nahe, dass diese Kombination von Algorithmus, Trainings- und Evaluationsdaten nicht funktioniert. Unter Umständen könnte der Algorithmus bessere Ergebnisse erzielen, wenn er ebenfalls mit dem Datensatz NRW *trainiert* würde, da Trainieren und Evaluieren mit dem Datensatz Kalifornien gut funktioniert hat. Wir hatten davon abgesehen, den Algorithmus mit dem Datensatz NRW zu trainieren, weil dieser zu klein war.

Die exakten Ergebnisse unserer Klassifizierungs-Algorithmen haben wir auf *GitLab* veröffentlicht, siehe [2].

8.4 Was man noch untersuchen könnte

Mit unserer Arbeit haben wir noch lange nicht alle Möglichkeiten betrachtet, mit denen man PV-Anlagen lokalisieren kann. In den folgenden Abschnitten werden ein paar Ideen vorgestellt, die in zukünftige Forschung einfließen können.

8.4.1 Ausblick zu Datenquellen

- Wir haben lediglich eine kleine Region untersucht (vgl. Abschnitt 4.2.1). Das ließe sich auf gesamt Deutschland ausweiten und eventuell auch auf andere Staaten. Zukünftige Arbeiten könnten Orthophotos von anderen Bundesländern mit einbeziehen. Wegen der in Abschnitt 4.2 beschriebenen Verschiedenheit von Verfügbarkeit, Qualität und Aktualität der Orthophotos muss die Vergleichbarkeit der damit gewonnenen Ergebnisse kritisch betrachtet werden. Zu den Orthophotos müssen dann auch die PV-Anlagen annotiert werden. Diese werden in größerer Quantität benötigt, was durch Crowdsourcing bewerkstelligt werden könnte.

- Wir haben keine Recherche zu von den Stromnetzbetreibern bereitgestellten Daten angestellt. Diese könnten vermutlich hilfreich bei der Lokalisierung von PV-Anlagen sein.
- Die Daten aus dem Marktstammdatenregister (Anzahl und kumulierte Größe von PV-Modulen pro Postleitzahl, evtl. auch die Modulorientierung) könnten nach der Lokalisierung von PV-Anlagen als zur groben Validierung herangezogen werden.
- Von Freiwilligen gepflegte Datenbanken zu Standorten von PV-Anlagen (z.B. *PVOutput*[34]) könnten wie das Marktstammdatenregister zur Validierung oder zur Erstellung von Datensätzen für weitere Lokalisierungs-Algorithmen evaluiert werden.
- Wir haben unseren Datensatz so beschnitten, dass wir nur Gebäudedächer analysieren. Da nicht jede PV-Anlage auf einem Gebäudedach angebracht ist, besteht noch Forschungsbedarf an der Lokalisierung solcher anderen PV-Anlagen.
- Es gibt viele wissenschaftliche Arbeiten, die sich mit dem Potential für PV-Anlagen beschäftigen (vgl. [24]). Die potentiell für PV-Anlagen nutzbare Fläche könnte zur Reduzierung des bei der Lokalisierung zu durchsuchenden Bereiches verwendet werden.

8.4.2 Ausblick zu Algorithmen

Wir haben uns mit unserem Algorithmus an bisheriger Forschung orientiert, ansonsten aber basierend auf Erfahrungen eine eigene experimentelle Architektur gewählt. Es besteht noch Forschungsbedarf an der Evaluation der Eignung verschiedener Prinzipien des maschinellen Lernens, Architekturen, Implementierungen und Kombinationen dieser für die Lokalisierung von PV-Anlagen. Ebenso könnten „klassische“ Algorithmen evaluiert werden. Es gibt bereits verschiedene erfolgreiche Ansätze, doch diese sind Aufgrund von verschiedenen Klassifizierungs-Auflösungen und Bewertungsmethoden nur

[34] vgl. `https://pvoutput.org/about.html` (abgerufen am 31.05.2020)

eingeschränkt vergleichbar. Zukünftige Forschung sollte versuchen, die Vergleichbarkeit der Ansätze herzustellen.

Das in anderen Arbeiten gefundene theoretische Potential für die Installation von PV-Modulen (vgl. [24]) könnte als zu durchsuchende Fläche für die Lokalisierung von PV-Modulen verwendet werden.

8.4.3 Sonstige Ideen

- In dem von uns annotierten Datensatz differenzieren wir nicht zwischen PV-Modulen und Solarkollektoren oder gar verschiedenen Bauarten jener (vgl. Abschnitt 1.1). Die Beachtung der Typen dürfte den Nutzen des Ergebnisses verbessern und die Beachtung der Bauarten könnte die Lokalisierung verbessern.

- Die Modulorientierung ist eine hilfreiche Information für die Schätzung der mit PV erzeugten Leistung (vgl. Abschnitt 3.1.1). Dafür gibt es bereits Ansätze basierend auf Orthophotos, Umringen und digitalen Höhenmodellen, doch durch die präzise Kenntnis der Standorte von PV-Modulen kann die Modulorientierung genauer bestimmt werden. Außerdem könnte die Abhängigkeit zwischen Modulorientierung und Dachorientierung genauer untersucht werden.

- Die anhand der Lokalisierung von PV-Module geschätzte mit PV erzeugte Leistung könnte mittels historischer Leistungsdaten auf Plausibilität geprüft werden.

- Ergebnis unseres Algorithmus' sind georeferenzierte Rastergrafiken, die geschätzte Standorte von Solarmodulen ausweisen. Zur leichteren Verwendung in darauf folgenden Auswertungen wäre es sicherlich hilfreich, die Rastergrafiken in georeferenzierte Umriss-Polygone umzurechnen. Damit wäre auch eine Anlagen-basierte Bewertung des Algorithmus möglich.

Literatur

[1] K. Bradbury, R. Saboo, J. Malof, T. Johnson, A. Devarajan, W. Zhang, L. Collins und R. Newell, *Distributed Solar Photovoltaic Array Location and Extent Data Set for Remote Sensing Object Identification*, Mai 2016. DOI: 10.6084/m9.figshare.3385780.v1. Adresse: https://figshare.com/articles/Distributed_Solar_Photovoltaic_Array_Location_and_Extent_Data_Set_for_Remote_Sensing_Object_Identification/3385780/1.

[2] S. Petrik und F. Hamme, *Lokalisierung von Solaranlagen mithilfe von Orthophotos und Kartendaten*, Ergebnisse und Datensatz, Juni 2020. Adresse: https://gitlab.com/dh-studienarbeit/pv-lokalisierung.

[3] D. H. Wirth, *Aktuelle Fakten zur Photovoltaik in Deutschland*, Fraunhofer-Institut für Solare Energiesysteme ISE, Feb. 2020. Adresse: https://www.ise.fraunhofer.de/content/dam/ise/de/documents/publications/studies/aktuelle-fakten-zur-photovoltaik-in-deutschland.pdf.

[4] S. Killinger, "Anlagenscharfe Simulation der PV-Leistung basierend auf Referenzmessungen und Geodaten," Diss., Karlsruher Institut für Technologie (KIT), 2018, 204 S., ISBN: 978-3-8396-1349-8. DOI: 10.5445/IR/1000082916. Adresse: https://www.bookshop.fraunhofer.de/buch/249167.

[5] Bundesministerium für Wirtschaft und Energie (BMWi), *Die Energie der Zukunft, Sechster Monitoring-Bericht zur Energiewende, Berichtsjahr 2016*, Juni 2018. Adresse: https://www.bmwi.de/Redaktion/DE/Publikationen/Energie/sechster-monitoring-bericht-zur-energiewende.pdf?__blob=publicationFile&v=39.

[6] M. A. Zimmer, Apr. 2016. Adresse: https://pixabay.com/images/id-1322810/.

[7] B. Rieke, *Flachkollektoren zur Warmwasserbereitung und Heizungsunterstützung*, Juli 2011. Adresse: https://commons.wikimedia.org/wiki/File:Kollektoranlage.JPG.

[8] H. Watter, *Regenerative Energiesysteme*. Springer Fachmedien Wiesbaden, 2019. DOI: 10.1007/978-3-658-23488-1.

[9] Amt für Geoinformation, Vermessungs- und Katasterwesen, *Geobasisdaten amtlich flächendeckend aktuell Ausgabe 2019/2020*, Juni 2019. Adresse: https://www.laiv-mv.de/Geoinformation/Raumbezug/SAPOS/?id=14899&processor=veroeff.

[10] Ministerium für Wirtschaft, Innovation, Digitalisierung und Energie des Landes Nordrhein-Westfalen, *Digitale Orthophotos NW*, Mai 2018. Adresse: https://open.nrw/dataset/56fb584b-10cf-4009-a405-0bef06bb3e00.

[11] Bezirksregierung Köln, *Digitale Orthophotos*, Feb. 2020. Adresse: https://www.bezreg-koeln.nrw.de/brk_internet/geobasis/luftbildinformationen/aktuell/digitale_orthophotos/index.html.

[12] Landesamt für Vermessung und Geoinformation Sachsen-Anhalt, *Digitale Orthophotos*, 2020. Adresse: https://www.lvermgeo.sachsen-anhalt.de/de/digitale_orthophotos.html.

[13] User:Pieter Kuiper, *OrthoPerspective.svg*, Okt. 2010. Adresse: https://upload.wikimedia.org/wikipedia/commons/5/51/OrthoPerspective.svg.

[14] Hessische Verwaltung für Bodenmanagement und Geoinformation, *Produktkatalog Heft 3 Luftbildprodukte*, Mai 2016. Adresse: https://www.gds.hessen.de/INTERSHOP/static/WFS/HLBG-Geodaten-Site/-/HLBG-Geodaten/-/downloads/Heft3_Luftbildprodukte.pdf.

[15] Niedersächsisches Ministerium für Umwelt, Energie, Bauen und Klimaschutz, *Digitale Orthophotos Niedersachsen (DOP20)*, Dez. 2018. Adresse: https://numis.niedersachsen.de/portal/search-detail.psml?cmd=doShowDocument&docuuid=17f9d557-37b8-4932-81e3-17f728f0dba3&plugid=/ingrid-group:iplug-csw-dsc-lgln.

[16] Landesamt für Vermessung und Geobasisinformation Rheinland-Pfalz, *Digitale Orthofotos (DOP) Produktbeschreibung*, Dez. 2016. Adresse: https://lvermgeo.rlp.de/fileadmin/lvermgeo/pdf/produktblaetter/ProduktbeschreibungRP_DOP.pdf.

[17] D. Rolnick, P. L. Donti, L. H. Kaack, K. Kochanski, A. Lacoste, K. Sankaran, A. S. Ross, N. Milojevic-Dupont, N. Jaques, A. Waldman-Brown, A. Luccioni, T. Maharaj, E. D. Sherwin, S. K. Mukkavilli, K. P. Kording, C. Gomes, A. Y. Ng, D. Hassabis, J. C. Platt, F. Creutzig, J. Chayes und Y. Bengio, "Tackling Climate Change with Machine Learning," *arXiv preprint arXiv:1906.05433*, 2019.

[18] A. Nischwitz, M. Fischer und P. Haberäcker, *Computergrafik und Bildverarbeitung - Alles für Studium und Praxis - Bildverarbeitungswerkzeuge, Beispiel-Software und interaktive Vorlesungen online verfügbar*. Berlin Heidelberg: Springer Science & Business Media, 2007, ISBN: 978-3-834-80186-9.

[19] J. Yu, Z. Wang, A. Majumdar und R. Rajagopal, "DeepSolar: A Machine Learning Framework to Efficiently Construct a Solar Deployment Database in the United States," *Joule*, Jg. 2, Nr. 12, S. 2605–2617, 2018.

[20] J. Yuan, H.-H. L. Yang, O. A. Omitaomu und B. L. Bhaduri, "Large-scale solar panel mapping from aerial images using deep convolutional networks," *2016 IEEE International Conference on Big Data (Big Data)*, S. 2703–2708, 2016.

[21] R. Castello, S. Roquette, M. Esguerra, A. Guerra und J.-L. Scartezzini, "Deep learning in the built environment: automatic detection of rooftop solar panels using Convolutional Neural Networks," *CISBAT 2019 | Climate Resilient Cities – Energy Efficiency & Renewables in the Digital Era*, Jg. 1343, S. 12 034, 2019.

[22] A. Krizhevsky, I. Sutskever und G. E. Hinton, "ImageNet classification with deep convolutional neural networks," *Communications of The ACM*, Jg. 60, Nr. 6, S. 84–90, 2017.

[23] S. Dotenco, M. Dalsass, L. Winkler, T. Würzner, C. Brabec, A. Maier und F. Gallwitz, "Automatic detection and analysis of photovoltaic modules in aerial infrared imagery," in *2016 IEEE Winter Conference on Applications of Computer Vision (WACV)*, IEEE, 2016, S. 1–9.

[24] K. Fath, "Technical and economic potential for photovoltaic systems on buildings," Englisch, Diss., Karlsruher Institut für Technologie (KIT), 2018, 283 S., ISBN: 978-3-7315-0787-1. DOI: 10.5445/KSP/1000081498.

[25] OpenStreetMap contributors, *Planet dump retrieved from https://planet.osm.org*, https://www.openstreetmap.org, 2020.

[26] Stabsstelle Geodatenmanagement, *LOD2 Daten von Freiburg*, 2016. Adresse: https://www.service-bw.de/organisationseinheit/-/sbw-oe/Stabsstelle+Geodatenmanagement+Stadt+Freiburg+im+Breisgau-6008924-organisationseinheit-0.

[27] K. Mainzer, S. Killinger, R. McKenna und W. Fichtner, "Assessment of rooftop photovoltaic potentials at the urban level using publicly available geodata and image recognition techniques," *Solar Energy*, Jg. 155, S. 561–573, 2017.

[28] W. Fichtner, R. McKenna, S. Killinger, D. Schlund und K. Mainzer, "Rooftop PV Potential Estimations: Automated Orthographic Satellite Image Recognition Based on Publicly Available Data," *32nd European Photovoltaic Solar Energy Conference and Exhibition (EU PVSEC), München, June 20-24, 2016. Proceedings on DVD*, S. 2930–2933, 2016.

[29] J. M. Malof, K. Bradbury, L. M. Collins und R. G. Newell, "Automatic detection of solar photovoltaic arrays in high resolution aerial imagery," *Applied Energy*, Jg. 183, S. 229–240, 2016, ISSN: 0306-2619. DOI: https://doi.org/10.1016/j.apenergy.2016.08.191. Adresse: http://www.sciencedirect.com/science/article/pii/S0306261916313009.

[30] J. Malof, R. Hou, L. Collins, K. Bradbury und R. Newell, "Automatic solar photovoltaic panel detection in satellite imagery," Nov. 2015, S. 1428–1431. DOI: 10.1109/ICRERA.2015.7418643.

[31] J. M. Malof, L. M. Collins, K. Bradbury und R. G. Newell, "A deep convolutional neural network and a random forest classifier for solar photovoltaic array detection in aerial imagery," in *2016 IEEE International Conference on Renewable Energy Research and Applications (ICRERA)*, 2016, S. 650–654.

[32] J. M. Malof, K. Bradbury, L. M. Collins, R. G. Newell, A. Serrano, H. Wu und S. Keene, "Image features for pixel-wise detection of solar photovoltaic arrays in aerial imagery using a random forest classifier," in *2016 IEEE International Conference on Renewable Energy Research and Applications (ICRERA)*, 2016, S. 799–803.

[33] J. Matas, O. Chum, M. Urban und T. Pajdla, "Robust wide-baseline stereo from maximally stable extremal regions," *Image and Vision Computing*, Jg. 22, Nr. 10, S. 761–767, 2004.

[34] C. Cortes und V. Vapnik, "Support-vector networks," *Machine Learning*, Jg. 20, Nr. 3, S. 273–297, Sep. 1995, ISSN: 1573-0565. Adresse: https://doi.org/10.1007/BF00994018.

[35] D. Comaniciu und P. Meer, "Mean shift: a robust approach toward feature space analysis," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Jg. 24, Nr. 5, S. 603–619, 2002.

[36] L. Breiman, "Random Forests," *Machine Learning*, Jg. 45, Nr. 1, S. 5–32, Okt. 2001, ISSN: 1573-0565. Adresse: https://doi.org/10.1023/A:1010933404324.

[37] N. Otsu, "A threshold selection method from gray level histograms," *IEEE Transactions on Systems, Man, and Cybernetics*, Jg. 9, Nr. 1, S. 62–66, 1979.

[38] J. Malik, S. Belongie, T. Leung und J. Shi, "Contour and Texture Analysis for Image Segmentation," *International Journal of Computer Vision*, Jg. 43, Nr. 1, S. 7–27, 2001.

[39] S. Puttemans, W. Van Ranst und T. Goedemé, "Detection of photovoltaic installations in RGB aerial imaging: a comparative study," eng, 2016. Adresse: https://lirias.kuleuven.be/retrieve/396013.

[40] ——, *EAVISE Solar Panel Dataset*, Sep. 2017. Adresse: https://www.eavise.be/SolarPanelDataset.

[41] P. Viola und M. Jones, "Rapid object detection using a boosted cascade of simple features," in *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, Bd. 1, 2001, S. 511–518.

[42] Y. Freund und R. E. Schapire, "A Short Introduction to Boosting," 1999.

[43] S. Liao, X. Zhu, Z. Lei, L. Zhang und S. Z. Li, "Learning multi-scale block local binary patterns for face recognition," in *ICB'07 Proceedings of the 2007 international conference on Advances in Biometrics*, 2007, S. 828–837.

[44] P. Dollar, R. Appel, S. Belongie und P. Perona, "Fast Feature Pyramids for Object Detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Jg. 36, Nr. 8, S. 1532–1545, 2014.

[45] J. Yuan, "Automatic Building Extraction in Aerial Scenes Using Convolutional Networks," *CoRR*, Jg. abs/1602.06564, 2016. arXiv: 1602.06564. Adresse: http://arxiv.org/abs/1602.06564.

[46] C. Elkan, "The foundations of cost-sensitive learning," in *International joint conference on artificial intelligence*, Lawrence Erlbaum Associates Ltd, Bd. 17, 2001, S. 973–978.

[47] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens und Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, S. 2818–2826.

[48] S. J. Pan und Q. Yang, "A Survey on Transfer Learning," *IEEE Transactions on Knowledge and Data Engineering*, Jg. 22, Nr. 10, S. 1345–1359, 2010.

[49] G. Hinton, N. Srivastava und K. Swersky, "Neural networks for machine learning lecture 6a overview of mini-batch gradient descent," *Cited on*, Jg. 14, Nr. 8, 2012.

[50] X. J. Zhu, "Semi-supervised learning literature survey," University of Wisconsin-Madison Department of Computer Sciences, Techn. Ber., 2005.

[51] O. Ronneberger, P. Fischer und T. Brox, "U-Net: Convolutional Networks for Biomedical Image Segmentation," *CoRR*, Jg. abs/1505.04597, 2015. arXiv: 1505.04597. Adresse: http://arxiv.org/abs/1505.04597.

[52] R. Gómez, *Understanding Categorical Cross-Entropy Loss, Binary Cross-Entropy Loss, Softmax Loss, Logistic Loss, Focal Loss and all those confusing names*, Mai 2018. Adresse: https://gombru.github.io/2018/05/23/cross_entropy_loss/.

[53] D. P. Kingma und J. Ba, *Adam: A Method for Stochastic Optimization*, 2014. eprint: arXiv:1412.6980.

[54] Bundesamt für Kartographie und Geodäsie, *Dokumentation Digitale Orthophotos DOP20 / DOP40*, Apr. 2019. Adresse: http://sg.geodatenzentrum.de/web_bkg_webmap/applications/info/dop.html.

[55] Landesamt für Geoinformation und Landentwicklung, *Digitale Orthophotos*. Adresse: https://www.lgl-bw.de/unsere-themen/Produkte/Geodaten/Digitale-Orthophotos/.

[56] Bayrische Vermessungsverwaltung, *Digitale Orthophotos*. Adresse: https://geodatenonline.bayern.de/geodatenonline/seiten/dop_info.

[57] Landesvermessung und Geobasisinformation Brandenburg, *Luftbilder aktuell*, 2020. Adresse: https://geobasis-bb.de/lgb/de/geodaten/luftbilder/luftbilder-aktuell/.

[58] Landesamt Geoinformation Bremen, *Luftbildprodukte*. Adresse: https://www.geo.bremen.de/produkte/luftbildprodukte-11703.

[59] Landesbetrieb Geoinformation und Vermessung, *Digitale Orthophotos*. Adresse: https://www.hamburg.de/bsw/geodaten/6702454/digitaleorthophotos.

[60] Behörde für Kultur und Medien, *Digitale Orthophotos Hamburg*, Jan. 2020. Adresse: http://suche.transparenz.hamburg.de/dataset/digitale-orthophotos-hamburg1?forceWeb=true.

[61] Hessische Verwaltung für Bodenmanagement und Geoinformation, *Digitale Orthophotos (ATKIS®-DOPs)*. Adresse: https://hvbg.hessen.de/sites/hvbg.hessen.de/files/DOP%20Produktblatt.pdf.

[62] ——, *Bildflugprogramm 2018 - 2019*, März 2018. Adresse: https://hvbg.hessen.de/sites/hvbg.hessen.de/files/Bildflugprogramm.pdf.

[63] Landesamt für Geoinformation und Landesvermessung Niedersachsen, *Digitale Orthophotos (ATKIS-DOP)*. Adresse: https://www.geobasisdaten.niedersachsen.de/shop/index.php?kat=DOP.

[64] Landesamt für Vermessung und Geobasisinformation Rheinland-Pfalz, *Digitale Orthophotos (DOP)*. Adresse: https://lvermgeo.rlp.de/de/produkte/geotopografie/luftbildprodukte/digitale-orthophotos/.

[65] ——, *Luftbild RP Basisdienst*, Digitale Orthophotos (DOP) - Dienst für entzerrte Luftbilder der Vermessungs- und Katasterverwaltung Rheinland-Pfalz mit einer Bodenauflösung von 40 cm. Adresse: https://www.geoportal.rlp.de/mapbender/php/mod_showMetadata.php?resource=wms&languageCode=de&id=1819.

[66] Zentrale Stelle GDI-SL, *SL-Netz: DOP20 (2019)*, Digitale Orthophotos 2019 Saarland, nur im Landesdatennetz verfügbar. Adresse: http://geoportal.saarland.de/mapbender/php/mod_showMetadata.php?resource=layer&id=42525.

[67] Staatsbetrieb Geobasisinformation und Vermessung Sachsen, *ADV-WMS-DE-SN-DOP-RGB*, Metadaten Portal, Nov. 2019. Adresse: https://geoportal.sachsen.de/cps/metadaten_portal.html?id=26df8686-08cd-4dc2-b459-4d51b9badfe8.

[68] ——, *Downloadbereich DOP*. Adresse: https://www.geodaten.sachsen.de/downloadbereich-dop-4158.html.

[69] ——, *Rechtsgrundlagen und Nutzungsbedingungen*. Adresse: https://www.geodaten.sachsen.de/rechtsgrundlagen-und-nutzungsbedingungen-4509.html.

[70] Der Ministerpräsident des Landes Schleswig-Holstein - Staatskanzlei, *ATKIS®-Digitale Orthophotos (DOP)*. Adresse: https://www.schleswig-holstein.de/DE/Landesregierung/LVERMGEOSH/Service/serviceGeobasisdaten/geodatenService_Geobasisdaten_DOP_digital.html.

[71] Thüringer Landesamt für Bodenmanagement und Geoinformation, *Web-Map-Services Geoproxy*, Jan. 2020. Adresse: https://www.gdi-de.org/gds/print.php?uuid=d9804083-023f-4169-b2fe-3a6c3f3815b8&lang=de.

[72] ——, *Orthophotos*. Adresse: https://www.thueringen.de/th9/tlbg/geoinformation/geotopographie/luftbilder/orthophotos/.

[73] Bundesnetzagentur, *MaStR-Daten registriert ab 31.01.2019 (Stand 03.03.2020)*, 2020. Adresse: https://www.bundesnetzagentur.de/DE/Sachgebiete/ElektrizitaetundGas/Unternehmen_Institutionen/DatenaustauschundMonitoring/Marktstammdatenregister/MaStR_node.html#doc514816bodyText4.

[74] Statistisches Bundesamt, *Städte (Alle Gemeinden mit Stadtrecht) nach Fläche, Bevölkerung und Bevölkerungsdichte am 31.12.2018*, Okt. 2019. Adresse: https://www.destatis.de/DE/Themen/Laender-Regionen/Regionales/Gemeindeverzeichnis/Administrativ/05-staedte.html.

[75] Bundesamt für Kartographie und Geodäsie, *Amtliche Hausumringe Deutschland (HU-DE)*, Apr. 2018. Adresse: https://gdz.bkg.bund.de/index.php/default/digitale-geodaten/sonstige-geodaten/amtliche-hausumringe-deutschland-hu-de.html.

[76] H. Fan, A. Zipf, Q. Fu und P. Neis, "Quality assessment for building footprints data on OpenStreetMap," *International Journal of Geographical Information Science*, Jg. 28, Nr. 4, S. 700–719, 2014. DOI: 10.1080/13658816.2013.867495. eprint: https://doi.org/10.1080/13658816.2013.867495. Adresse: https://doi.org/10.1080/13658816.2013.867495.

[77] J. J. Arsanjani, P. Mooney, A. Zipf und A. Schauss, "Quality Assessment of the Contributed Land Use Information from OpenStreetMap Versus Authoritative Datasets," *OpenStreetMap in GIScience*, S. 37–58, 2015.

[78] Microsoft Corporation, *Computer generated building footprints for the United States*, GitHub Repository, Sep. 2019. Adresse: https://github.com/microsoft/USBuildingFootprints/tree/e6cc9c78f209066ea824524c3fdb64326af06ad8.

[79] K. Wada, *labelme: Image Polygonal Annotation with Python*, https://github.com/wkentaro/labelme, 2016.

[80] G. Bradski, "The OpenCV Library," *Dr. Dobb's Journal of Software Tools*, 2000.

[81] S. van der Walt, S. C. Colbert und G. Varoquaux, "The NumPy Array: A Structure for Efficient Numerical Computation," *Computing in Science Engineering*, Jg. 13, Nr. 2, S. 22–30, März 2011, ISSN: 1558-366X. DOI: 10.1109/MCSE.2011.37.

[82] T. E. Oliphant, "Guide to NumPy," 2015.

[83] S. Gillies u. a., *Shapely: manipulation and analysis of geometric objects*, toblerity.org, 2007–. Adresse: https://github.com/Toblerity/Shapely.

[84] A. Clark, *Pillow (PIL Fork) Documentation*, Release 7.2.0.dev0, Juni 2020. Adresse: https://buildmedia.readthedocs.org/media/pdf/pillow/latest/pillow.pdf.

[85] B. Steiner, Z. DeVito, S. Chintala, S. Gross, A. Paszke, F. Massa, A. Lerer, G. Chanan, Z. Lin, E. Yang, A. Desmaison, A. Tejani, A. Kopf, J. Bradbury, L. Antiga, M. Raison, N. Gimelshein, S. Chilamkurthy, T. Killeen, L. Fang und J. Bai, "PyTorch: An Imperative Style, High-Performance Deep Learning Library," in *NeurIPS 2019 : Thirty-third Conference on Neural Information Processing Systems*, 2019, S. 8026–8037.

[86] A. Zhou, B.-Y. Qu, H. Li, S.-Z. Zhao, P. N. Suganthan und Q. Zhang, "Multiobjective evolutionary algorithms: A survey of the state of the art," *Swarm and evolutionary computation*, Jg. 1, Nr. 1, S. 32–49, 2011.

[87] User:Berklas, *Pareto Optimum*, Dez. 2008. Adresse: https://commons.wikimedia.org/wiki/File:Pareto-optasoarp.svg.

[88] The ImageMagick Development Team, *ImageMagick*, Version 7.0.10, 12. Juni 2020. Adresse: https://imagemagick.org.

[89] T. Esch, W. Heldens und A. Metz, "Die Erde im Bild – Satelliten als Werkzeug zur Beobachtung der Landoberfläche," in *Globale Urbanisierung: Perspektive aus dem All*, H. Taubenböck, M. Wurm, T. Esch und S. Dech, Hrsg. Berlin, Heidelberg: Springer Berlin Heidelberg, 2015, S. 23–27, ISBN: 978-3-662-44841-0. DOI: 10.1007/978-3-662-44841-0_4. Adresse: https://doi.org/10.1007/978-3-662-44841-0_4.

[90] A. Shelestov, M. Lavreniuk, N. Kussul, A. Novikov und S. Skakun, "Exploring Google Earth Engine Platform for Big Data Processing: Classification of Multi-Temporal Satellite Imagery for Crop Mapping," *Frontiers in Earth Science*, Jg. 5, S. 17, 2017, ISSN: 2296-6463. DOI: 10.3389/feart.2017.00017. Adresse: https://www.frontiersin.org/article/10.3389/feart.2017.00017.

[91] J. Silva Centeno, "Integrierte Verarbeitung von Satellitenbild- und gescannter Karteninformation [online]," Diss., 2000. DOI: 10.5445/IR/4262000.

[92] M. Kapoor und R. D. Garg, "Cloud computing for energy requirement and solar potential assessment," *Spatial Information Research*, Jg. 26, Nr. 4, S. 369–379, Aug. 2018, ISSN: 2366-3294. Adresse: https://doi.org/10.1007/s41324-018-0181-3.

[93] N. Gorelick, M. Hancher, M. Dixon, S. Ilyushchenko, D. Thau und R. Moore, "Google Earth Engine: Planetary-scale geospatial analysis for everyone," *Remote Sensing of Environment*, Jg. 202, S. 18–27, 2017, Big Remotely Sensed Data: tools, applications and experiences, ISSN: 0034-4257. DOI:

https://doi.org/10.1016/j.rse.2017.06.031. Adresse: http://www.sciencedirect.com/science/article/pii/S0034425717302900.

[94] V. Quaschning, *Systemtechnik einer klimaverträglichen Elektrizitätsversorgung in Deutschland für das 21. Jahrhundert*, Ser. Fortschritt-Berichte VDI: Reihe 6, Energietechnik, Wärmetechnik. VDI-Verlag, 2000, ISBN: 9783183437061. Adresse: https://books.google.de/books?id=oB%5C_jAAAACAAJ.

[95] Bezirksregierung Köln, Abteilung 7 / Geobasis NRW, *Digitale Orthophotos NW*, https://open.nrw/dataset/56fb584b-10cf-4009-a405-0bef06bb3e00, Mai 2018.

[96] P. Mukhopadhyay und B. B. Chaudhuri, "A survey of Hough Transform," *Pattern Recognition*, Jg. 48, Nr. 3, S. 993–1010, 2015.

[97] J. Canny, "A computational approach to edge detection," *IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE*, 1986. DOI: 10.1109/TPAMI.1986.4767851.