Maschinelles Lernen für Physiker*innen

Projektbericht

Segmentierung von Gewässern auf Satellitenbildern

Nico Guth
nico.guth@tu-dortmund.de

Projektpartner: Samuel Haefs

samuel.haefs@tu-dortmund.de

Abgabe: 15.08.2021

TU Dortmund – Fakultät Physik

Inhaltsverzeichnis

1 Einleitung und Zielsetzung

Schon lange ist die Kartografie ein wesentlicher Teil des menschlichen Fortschritts. Gegen Ende des 20. Jahrhunderts gewann die Kartografie anhand von Satellitenbildern zunehmend an Wichtigkeit. Dadurch hatten und haben Kartografen einen deutlich geringeren Aufwand große Gebiete der Erde abzudecken. Da allerdings die zu kartografierende Fläche auf der Erde sehr groß ist, scheint eine vollständig manuelle Kartografie nicht mehr sinnvoll zu sein. Ein zusätzliches Problem ist, dass Karten durchgehend angepasst werden müssen, da die Welt kontinuierlich im Wandel ist, sowohl durch menschliches Einwirken als auch durch natürliche Phänomene.

Um das Erstellen von Karten deutlich zu beschleunigen, wird seit einigen Jahren die Analyse der Satellitenbilder hauptsächlich von Computern durchgeführt. Hierfür eignet sich maschinelles Lernen und insbesondere das „Deep Learning“ mit neuronalen Netzwerken. Aber auch wenn heutzutage große Konzerne wie Google, Apple oder Microsoft schon sehr gute Algorithmen entwickelt haben, ist es immer noch deutlich zu erkennen, dass auch diese Algorithmen noch Fehler machen.

Fragestellung: Wo befinden sich auf einem gegebenen Satellitenbild innerhalb Europas größere Wasserflächen?

Ziel dieses Projektes ist also die Erkennung von Gewässern auf Satellitenbildern. Genauer gesagt wird mithilfe von maschinellem Lernen zu jedem Eingangsbild eine Maske erzeugt, die jedem Pixel entweder Wasser oder kein Wasser zuordnet. Außerdem werden nur Satellitenbilder innerhalb europäischer Länder und auf einer bestimmten Zoomstufe betrachtet. Dadurch können nur größere Gewässer, also Seen, Flüsse, das Meer oder ähnliches erkannt werden und kleinere Gewässer wie Bäche, Teiche oder ähnliches sind nicht von großer Bedeutung für dieses Projekt. Weitere Details zum Datensatz sind in Abschnitt 2 erläutert.

Das Projekt ist Teil des Seminars „Maschinelles Lernen für Physiker*innen“ im Sommersemester 2021 und wurde von Samuel Haefs und mir (Nico Guth) durchgeführt.

2 Datensatz

Da kein passender Datensatz gefunden wurde, musste dieser auch als Teil des Projekts erstellt werden. Das Ziel war es für viele verschiedene Orte innerhalb europäischer Grenzen ein Satellitenbild und eine passende Maske zu sammeln. Die Maske soll dabei die Pixel markieren, die zu größeren Wasserflächen wie Seen, Teichen, Flüssen, Kanälen oder Küstenabschnitten gehören. Zusätzlich werden zu jedem Ort einige Metadaten wie die Koordinaten und das zugehörige Land in einer CSV Datei gespeichert. Ein Beispiel vom erzeugten Datensatz ist in Abbildung 1 zu sehen.

(a) Satellitenbild **(b)** Maskenbild

Abbildung 1: Beispiel des Datensatzes an den Koordinaten (in Deutschland) $x_{\text{Tile}} = 16998$, $y_{\text{Tile}} = 10927$, $z_{\text{Tile}} = 15$, $x_{\text{Geo}} = 6,751°$, $y_{\text{Geo}} = 51,293°$. ©Mapbox, ©OpenStreetMap

Die Datensatz-Erstellung wurde mit Python automatisiert und wird im Folgenden erläutert.

Die Daten wurden nicht in einer Sitzung, sondern immer in kleinen Paketen gesammelt. Dafür wurden zuerst gleichverteilte Zufallszahlen für Längengrad x_{Geo} und Breitengrad y_{Geo} (also in geographischen Koordinaten) im Bereich $x_{\text{Geo}} \in [-10,49°\,,\,40,27°]$, $y_{\text{Geo}} \in [34,51°\,,\,71,20°]$ gezogen. Dieser Bereich ist eine grobe Begrenzung für Europa. Für jeden so gezogenen Ort wurde mithilfe der groben Ländergrenzen[1] des „Natural Earth“ Projektes überprüft, ob dieser Ort innerhalb eines europäischen Landes liegt. [1] Hierbei wurden Russland und Island nicht miteinbezogen.

Ziel war es viele Satellitenbilder zu sammeln, die Wasser enthalten. Da in Europa allerdings das Verhältnis von Wasser- zu Landfläche sehr gering ist, musste vor dem

[1] `https://www.naturalearthdata.com/downloads/110m-cultural-vectors/110m-admin-0-countries/`

Herunterladen der Wassergehalt des Bildes überprüft werden. Dies wurde mithilfe der Daten der „Mapbox Vector Tiles API“[2] überprüft. [2] Ein Ort wurde nur dann akzeptiert, wenn dieser nicht schon zuvor gezogen wurde und das Bild sowohl einen Wasseranteil als auch einen Landanteil hat.

Für jeden akzeptierten Ort wurde dann mithilfe der „Mapbox Raster Tiles API“[2] ein Satellitenbild heruntergeladen. Außerdem wurde von diesem Ort mithilfe der „Mapbox Static Tiles API“[2] und einem selbst angepassten Kartenstil ein Maskenbild für den Wassergehalt heruntergeladen.[2] Mapbox verwendet die Kartendaten von „OpenStreetMap“ und somit stammen auch die Daten der Maskenbilder daher.[3]

An dieser Stelle scheint es sinnvoll zu erwähnen, dass die Maskenbilder teilweise Fehler enthalten und Gewässer nur sehr grob oder auch falsch zugeordnet sind.

In der Bereitstellung von digitalen Karten und geographischen Daten wird heutzutage vermehrt das Konzept sogenannter „Tiles“ verwendet.[4] Hier ist eine Zoom-Stufe z_{Tile} anzugeben und die Welt wird in $4^{z_{\text{Tile}}}$ gleichgroße Quadrate(„Tiles“) aufgeteilt. Jedem dieser Quadrate wird eine x_{Tile}- und eine y_{Tile}-Koordinate zugeordnet.

Während der Erstellung des Datensatzes musste zwischen einigen Koordinatensystemen gewechselt werden, aber die Koordinaten der „Tiles“ waren ausschlaggebend für die heruntergeladenen Bilder. Jedes Bild im Datensatz wurde mit einer Zoom-Stufe $z_{\text{Tile}} = 15$ generiert und entspricht genau einem „Tile“.

Der Datensatz enthielt am Ende des Projektes 57931 Satellitenbilder und dazu passende Maskenbilder. Jedes Bild hatte eine Auflösung von 256 x 256 Pixel.

3 Lösungsansatz

Ziel dieses Projektes war es Wasser auf Satellitenbildern zu erkennen. Dazu wurden zwei Methoden des maschinellen Lernens für die Segmentierung verwendet. Ein „Convolutional Neural Network“ stellte dabei die Hauptmethode dar und ein „Random Forest“ stellte dabei eine Alternativmethode zum Vergleich dar. Für beide Methoden wurde das Framework „TensorFlow“ verwendet.[5] Die Metrik, die benutzt wurde, um

[2] `https://docs.mapbox.com/api/maps/`

die Methoden zu evaluieren, war die Genauigkeit (zu Englisch „accuracy“), also der Anteil der richtig zugeordneten Pixel.

3.1 Vorverarbeitung

Der in Abschnitt 2 beschriebene Datensatz wurde in drei Teildatensätze aufgeteilt, den Trainings-, den Validierungs- und den Testdatensatz. Die Größen dieser Datensätze sind in Tabelle 1 angegeben.

Tabelle 1: Größen der Teildatensätze vom Gesamtdatensatz mit 57931 Satellitenbildern.

	Trainingsdaten	Validierungsdaten	Testdaten
Anzahl	39392	6952	11587
Anteil	68%	12%	20%

Außerdem wurden die Bilder zu einer Auflösung von 128 x 128 Pixel skaliert, da so nicht viel Information verloren ging und viel Rechenzeit und Rechenleistung eingespart werden konnte. Die Pixel der Maskenbilder wurden zu binären Werten umgerechnet, wobei 1 Wasser und 0 kein Wasser kennzeichnet und die drei Farbkanäle Satellitenbilder wurden auf einen Pixelwert zwischen 0 und 1 normiert.

Alle Datensätze wurden in „Batches“ aufgeteilt und aus Arbeitsspeichergründen während des Abrufens in den Arbeitsspeicher geladen. Dafür wurde eine „Batch“-Größe von 128 Satellitenbildern verwendet.

3.2 Hauptmethode: „Convolutional Neural Network“

Das verwendete neuronale Netzwerk ist vollständig „convolutional“ und orientiert sich an der häufig für Segmentierung verwendeten „U-Net“-Struktur.[6] Ein Beispiel eines solchen „U-Net“ ist in Abbildung 2 dargestellt. Hier wird in der ersten Hälfte des Netzwerks zwischen den „convolutional“ Lagen das Bild durch „maximales Pooling“ verkleinert (halbierte Kantenlänge) und in der zweiten Hälfte das Bild zwischen den „convolutional“ Lagen durch transponiert-„convolutional“ Lagen vergrößert (verdoppelte Kantenlänge). Zusätzlich wird die Ausgabe vor dem Verkleinern kopiert und mit der

Ausgabe nach dem Vergrößern verkettet. Dadurch entsteht die charakteristische U-Form, dargestellt mit mehreren Ebenen in der Höhe.

Allerdings unterscheidet sich das verwendete Netzwerk in einigen Hyperparametern, wie z.B. den Filteranzahlen, den Pixelanzahlen und der Tiefe zu dem gezeigten „U-Net“. Außerdem wurde zwischen den beiden „convolutional“ Lagen einer Ebene das Netzwerk mithilfe von „Dropout“ regularisiert. Ein weiterer Unterschied ist, dass bei den „convolutional“ Lagen ein „Padding“ von „same“ statt „valid“ verwendet wurde.

Als Aktivierungsfunktion wurde für alle „convolutional“ Lagen die „ReLU“-Funktion und für die letzte Lage die „Sigmoid“-Funktion verwendet. Der Optimierer „Adam“ hat dabei die Verlustfunktion der binären Kreuzentropie minimiert.

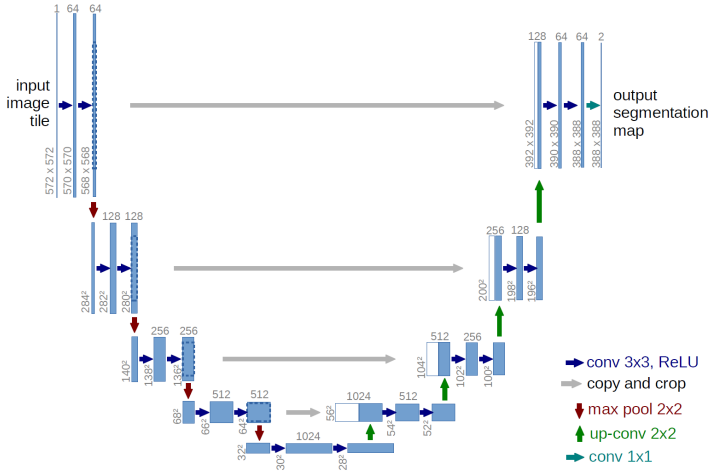


Abbildung 2: Visualisierung eines „U-Net“.[6]

Die besten Hyperparameter und die beste Struktur des Netzwerks wurde mithilfe einer zufälligen Gittersuche ermittelt. Das Netzwerk wurde für 100 verschiedene Kombinationen der Hyperparameter auf einem kleinen Trainingsdatensatz von 3000 Satellitenbildern für 20 Epochen trainiert und dann auf einem kleinen Validierungsdatensatz von ebenfalls 3000 Satellitenbildern evaluiert. Das Ergebnis der Hyperparametersuche ist in Abbildung 3 dargestellt und in Tabelle 2 sind die besten fünf Kombinationen aufgelistet. Vor dieser systematischen Suche wurden weitere Hyperparameter variiert, allerdings erwies sich das zuvor beschriebene neuronale Netzwerk als am besten geeignet und die systematische Suche beschränkte sich auf die wichtigsten Parameter. Variiert wurden folgende Hyperparameter:

- „filter_start“: Anzahl der Filter der „convolutional“ Lagen der obersten Ebene, wobei die Filter pro Ebene verdoppelt wurden

- „filter_levels“: Anzahl der Ebenen
- „kernel_size“: Größe der Faltungsmatrizen aller „convolutional“ Lagen
- „kernel_initializer“: Methode der Matrixinitialisierung aller „convolutional“ Lagen
- „dropout_start“: Stärke des „Dropout“ der obersten Ebene, wobei die Stärke jeweils nach zwei Ebenen um 0,1 erhöht wurde
- „learning_rate“: Lernrate des verwendeten Optimierers „Adam“

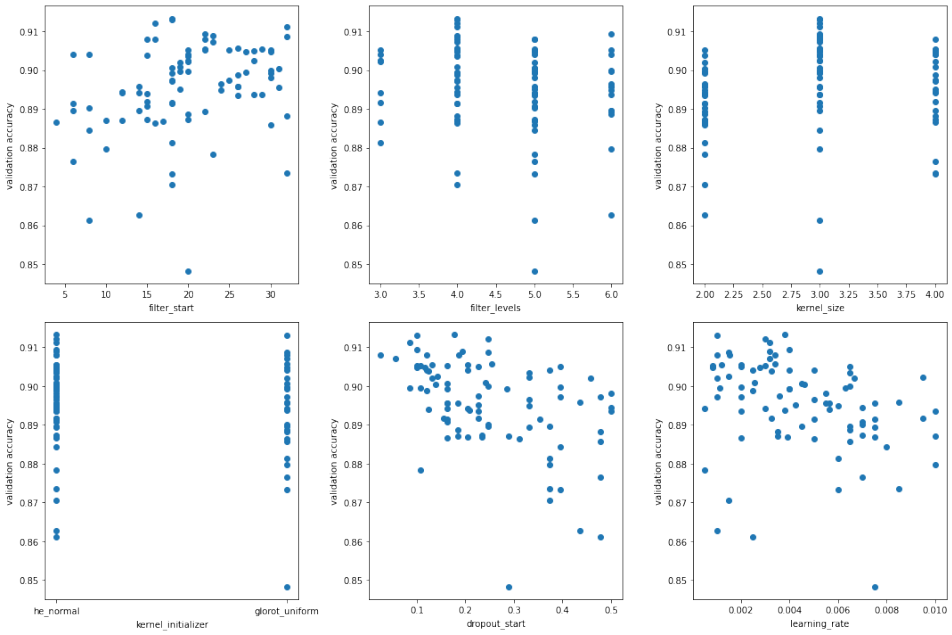


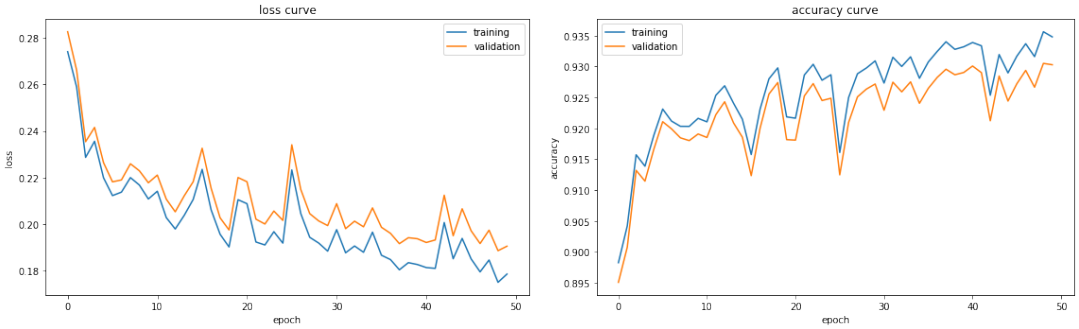
Abbildung 3: Zufällige Suche nach den besten Hyperparametern

Tabelle 2: Die besten fünf Ergebnisse der zufälligen Gittersuche mit der Genauigkeit auf den 3000 Testdaten und den zugehörigen Hyperparametern

Genauigkeit	„filter_start“	„filter_levels“	„kernel_size“	„kernel_initializer“	„dropout_start“	„learning_rate“
0.913355	18	4	3	he_normal	0.177895	0.003795
0.913100	18	4	3	glorot_uniform	0.100000	0.001000
0.912037	16	4	3	he_normal	0.247368	0.003000
0.911288	32	4	3	he_normal	0.086316	0.003179
0.909450	22	6	3	he_normal	0.100000	0.004000

Erkennbar sind Trends in „dropout_start“ und „learning_rate“ zu niedrigen Werten. Die Parameter „kernel_size“=3 und „filter_levels“=4 scheinen ebenfalls gut zu sein. „filter_start“ und „kernel_initializer“ zeigen keine deutlichen Trends. Aus der Suche ergab sich eine Kombination mit der besten Genauigkeit von 91,33% auf dem kleinen Testdatensatz. Das dadurch entstandene Netzwerk wurde dann für den gesamten Datensatz

verwendet und wird im Anhang anhand einer Visualisierung detailliert beschrieben. (siehe Abbildung 10)



(a) Verlustfunktion **(b)** Genauigkeit

Abbildung 4: Plots der Verlustfunktion bzw. Genauigkeit während dem Training je nach Epoche auf Trainings- und Validierungsdatensatz

Das Modell wurde mehrfach auf dem Trainingsdatensatz trainiert. Der beste Durchlauf nach 50 Epochen ist in Abbildung 4 dargestellt. Hier ist zu erkennen, dass das Modell etwas besser auf den Trainingsdaten abschnitt, allerdings ist nicht von einer Überanpassung auszugehen. Bei fast 40000 Bildern mit einer Auflösung von 128 x 128 Pixel und vielen deutlich unterschiedlichen Bildern ist eine Überanpassung nicht möglich und ein niedriger „Dropout“ wie hier reicht für die Regularisierung aus. Die Hyperparametersuche zeigte auch, dass ein zu hoher Dropout nur zu schlechteren Ergebnissen führte. Die „Zackigkeit“ der Kurven in Abbildung 4 lässt sich dadurch erklären, dass eine „Batch“-Größe von 128 Bildern verwendet wurde und ein großer Teil des Trainings so gar nicht in den Plots zu erkennen ist. Eine Erhöhung der Epochenanzahl war aufgrund von fehlenden Rechenkapazitäten nicht möglich.

Da die Pixel der Ausgabebilder des Netzwerks Werte zwischen 0 und 1 annimmt, wurde die Genauigkeit auf dem Validierungsdatensatz für verschiedene Schwellenwerte, ab denen ein Pixel als Wasser gewertet wird, berechnet und der beste Schwellenwert wurde verwendet. Hier ergab sich als bester Schwellenwert 0,54. Dies ist in Abbildung 5 dargestellt.

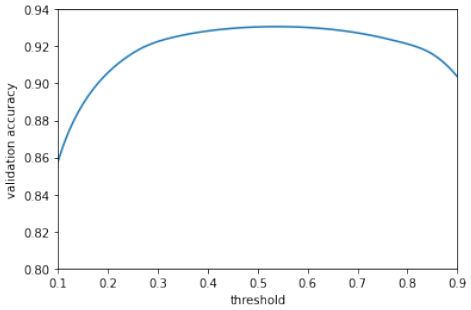


Abbildung 5: Plot der Genauigkeit auf dem Validierungsdatensatz gegen den Schwellenwert

3.3 Alternativmethode: „Random Forest“

Zum Vergleich mit dem neuronalen Netzwerk wurde eine pixelweise Klassifizierung mithilfe eines „Random Forest“ durchgeführt. Dieser „Random Forest“ bestand aus 60 Entscheidungsbäumen, die eine maximale Tiefe von 32 hatten. Auch hierfür wurde das Framework „TensorFlow“[3] verwendet.[5]

Hierfür wurden zunächst jedem Pixel der Satellitenbilder einige Attribute extrahiert. Auf jedem Farbkanal wurde der Betrag des Gradienten berechnet. Dann wurde vom Satellitenbild und dem Bild der Gradienten ein Graustufen Bild erzeugt. All diese 8 Attribute wurden dem „Random Forest“ pixelweise übergeben.

Aufgrund von Arbeitsspeicher- und Rechenzeit-Limitierungen konnte nicht der gesamte Trainingsdatensatz verwendet werden. Aus jedem Bild im Trainingsdatensatz wurden 25 zufällige Pixel verwendet und so wurde der „Random Forest“ auf etwa einer Millionen Pixel trainiert. Der Validierungs- und Testdatensatz blieb jedoch gleich.

4 Ergebnisse

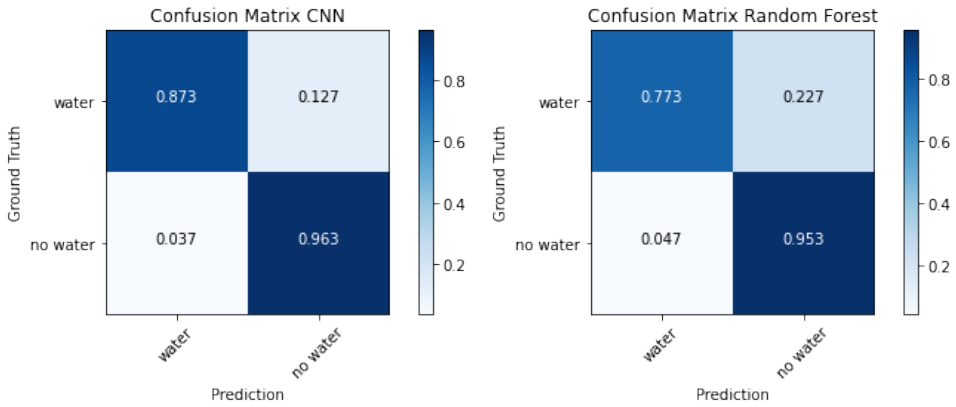
Nachdem beide Methoden trainiert wurden, wurden die so entstandenen Modelle evaluiert. In Tabelle 3 sind alle erreichten Genauigkeiten aufgelistet.

Tabelle 3: Genauigkeit der verwendeten Methoden auf den verschiedenen Teildatensätzen, wobei mit CNN das Convolutional Neural Network gemeint ist

Methode	Trainingsdaten	Validierungsdaten	Testdaten
CNN (standard Schwellenwert 0,50)	93,56%	93,05%	93,30%
CNN (bester Schwellenwert 0,54)	93,58%	93,07%	93,32%
Random Forest	89,40%	89,09%	89,37%

Hier ist deutlich zu erkennen, dass beide von uns gewählten Methoden gute Ergebnisse erzielten, jedoch die Hauptmethode eine deutlich höhere Genauigkeit erreichte. Eine genauere Analyse der Ergebnisse ist über eine Konfusionsmatrix erreichbar, welche in Abbildung 6 dargestellt ist. Auch hier bestätigt sich dass das neuronale Netz bessere Ergebnisse erzielte. Es ist abzulesen, dass als Wasser klassifizierte Pixel prozentual mehr Fehler enthalten. Allerdings ist anzumerken, dass nur etwa 33% der im Datensatz enthaltenen Pixel als Wasser gekennzeichnet sind.

[3] `https://www.tensorflow.org/decision_forests/`



(a) Hauptmethode **(b)** Alternativmethode

Abbildung 6: Konfusionsmatrizen der Vorhersagen der verwendeten Methoden auf dem Testdatensatz

Um die Methoden allerdings tatsächlich vergleichen zu können ist es notwendig einige Beispiele der vorhergesagten Masken zu betrachten. Dazu ist in Abbildung 7 die Ausgabe des Convolutional Neural Network mit und ohne binäre Darstellung mittels des Schwellenwertes dargestellt. Außerdem ist in Abbildung 8 die zusätzliche Eingabe des Gradienten des Satellitenbildes und die Ausgabe des Random Forest zu sehen.

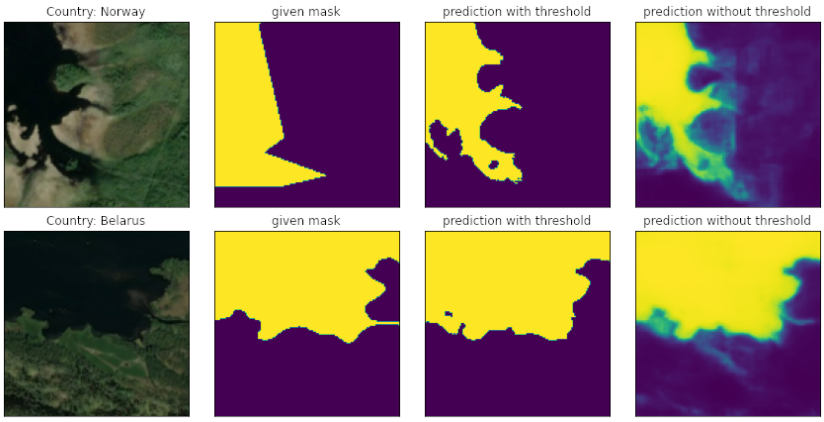


Abbildung 7: Beispiele der Vorhersagen des Convolutional Neural Network mit und ohne Anwendung des Schwellenwertes. ©Mapbox, ©OpenStreetMap

Schon hier lässt sich erkennen, dass die Alternativmethode deutlich mehr Rauschen aufgrund der pixelweisen Klassifizierung erzeugte. Die Hauptmethode konnte nicht alle Pixel eindeutig zuordnen und vor allem Ränder oder schwer zu klassifizierende Stellen erhielten Werte die eher mittig zwischen 0 und 1 liegen. Auch ist schon zu sehen, dass die gegebene Maske nicht perfekt war und teilweise Gewässer gar nicht

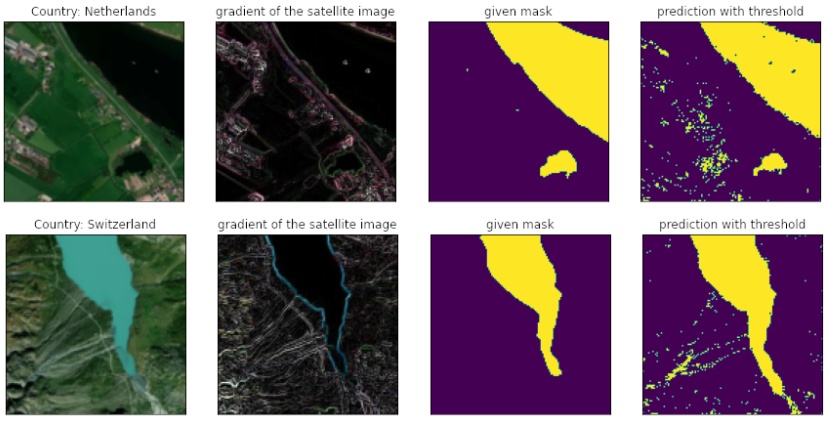


Abbildung 8: Beispiele der Vorhersagen des Random Forest und das Bild der Gradienten des Satellitenbildes. ©Mapbox, ©OpenStreetMap

oder nur grob markiert waren.

Um weitere Erkenntnisse zu den beiden Methoden zu bekommen, werden in Abbildung 9 die Vorhersagen beider Methoden miteinander verglichen. Weitere Beispiele sind im Anhang in Abbildung 11 und in Abbildung 12 dargestellt.

Häufig ist die gegebene Maske nicht sehr detailliert und obwohl beide Modelle auf diesen fehlerhaften Daten trainiert wurden, wurden die Grenzen der Gewässer häufig klar erkannt. Aufgrund dieser fehlerhaften gegebenen Masken war auch eine Genauigkeit von 100% nicht erreichbar.

In Abbildung 12 sind zusätzlich noch einige Beispiele aufgezeigt, bei denen die Segmentierung nicht gut funktionierte oder die gegebene Maske falsch war. Probleme waren z.B. Wolken, Ebbe und Flut, Felder, Sumpfgebiete, Schnee oder sonstige auch für den Menschen schwer zu erkennende Gewässer. Um tatsächlich alle Probleme sehen zu können, müssten deutlich mehr Bilder gezeigt werden, dies überschreitet allerdings den Rahmen dieses Berichtes. Teilweise ist nicht erklärbar warum die Algorithmen Wasser falsch erkannten. Häufig entstanden Fehler an Stellen, die auch ein Mensch nicht eindeutig klassifizieren könnte, aber manchmal wurden selbst eindeutige Kanten falsch zugeordnet. Einige der Fehler waren sicherlich auf die niedrige Auflösung von 128x128 Pixel zurückzuführen, aber die meisten Gewässer waren auch mit dieser Auflösung klar zu erkennen.

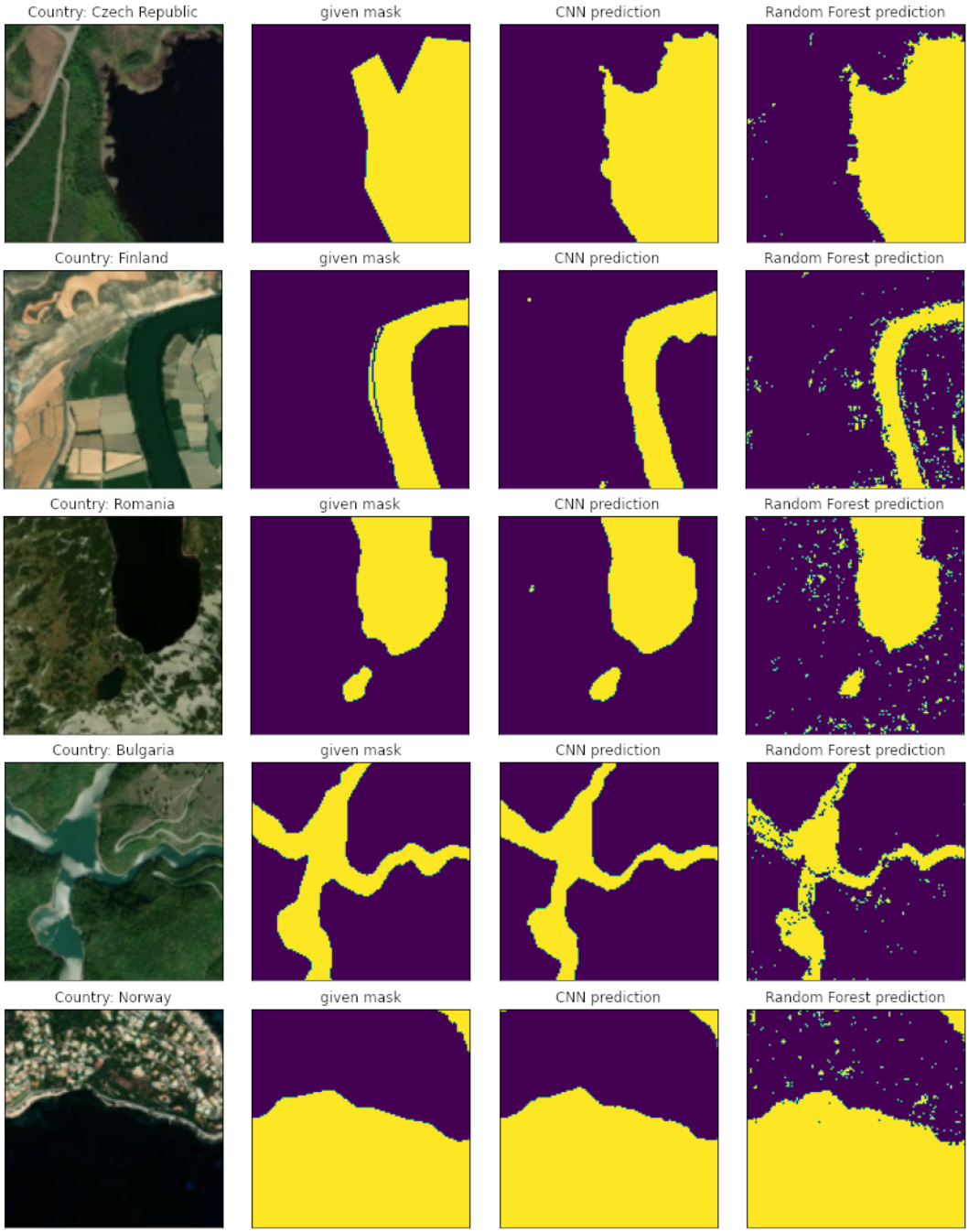


Abbildung 9: Beispiele zum Vergleich der Hauptmethode zur Alternativmethode. Aufgezeigt sind (von rechts nach links) das Satellitenbild, die Maske im Datensatz, die Vorhersage des Convolutional Neural Network und die Vorhersage des Random Forest.©Mapbox, ©OpenStreetMap

5 Fazit

Die Ergebnisse beider verwendeten Methoden zur Segmentierung von Gewässern auf Satellitenbildern waren ausreichend und übertrafen teilweise unsere Erwartungen vor dem Projekt. Erst während des Projektes erkannten wir, dass einige der Masken im Datensatz fehlerhaft waren. Auch erkannten wir, dass ein paar der Satellitenbilder für eine Segmentierung unbrauchbar waren. (z.B. Bilder von Wolken) Trotz der schlechten Daten, die zum Training verwendet wurden, wurden gute Ergebnisse erzielt und die vorhergesagten Masken waren häufig besser als die Masken des Datensatzes. Die erzielten Genauigkeiten der Hauptmethode von etwa 93% und der Alternativmethode von etwa 89% lassen sich allerdings nicht näher beurteilen, da nicht klar ist welche Genauigkeit die Masken im Datensatz erreichen und diese Masken als Grundwahrheit angenommen wurden. Hierfür wäre eine manuelle Überprüfung notwendig.

Der Vergleich des Convolutional Neural Network zum Random Forest zeigte deutlich, dass für dieses Problem ersteres besser geeignet war. Zusätzlich zu den besseren Ergebnissen war das Convolutional Neural Network deutlich sparsamer mit der benötigten Rechenleistung und sogar die Vorhersagen wurden deutlich schneller erzeugt als bei dem Random Forest. Auch ist zu sehen, dass das Convolutional Neural Network Beziehungen zwischen den Pixeln auswertete, was bei dem Random Forest nicht möglich war.

Durch eine ausführlichere Hyperparametersuche könnte das Modell weiter optimiert werden. Es wurde allerdings gezeigt, dass ein so komplexes und vielseitiges Problem wie die Wassererkennung auf Satellitenbildern mit Deep Learning gelöst werden kann. Eine Überwachung der Ausgabe von neuronalen Netzen durch Menschen ist allerdings noch zu empfehlen.

Abschließend sei zu sagen, dass dieses Projekt uns den Umgang mit Deep Learning weiter nahegebracht hat und gezeigt hat wie vielseitig einsetzbar neuronale Netzwerke sind.

Literatur

[1] Natural Earth. *Free vector and raster map data at 1:10m, 1:50m, and 1:110m scales*. 2021.
URL: https://www.naturalearthdata.com/.

[2] Mapbox. *Maps and location for developers*. 2021.
URL: https://www.mapbox.com/.

[3] OpenStreetMap. 2021.
URL: https://www.openstreetmap.org/.

[4] maptiler. *Tiles à la Google Maps*. 2021.
URL: https://www.maptiler.com/google-maps-coordinates-tile-bounds-projection/.

[5] Martín Abadi u. a. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. 2021.
URL: https://www.tensorflow.org/.

[6] Olaf Ronneberger, Philipp Fischer und Thomas Brox. *U-Net: Convolutional Networks for Biomedical Image Segmentation*. 2015. arXiv: 1505.04597 [cs.CV].

Anhang

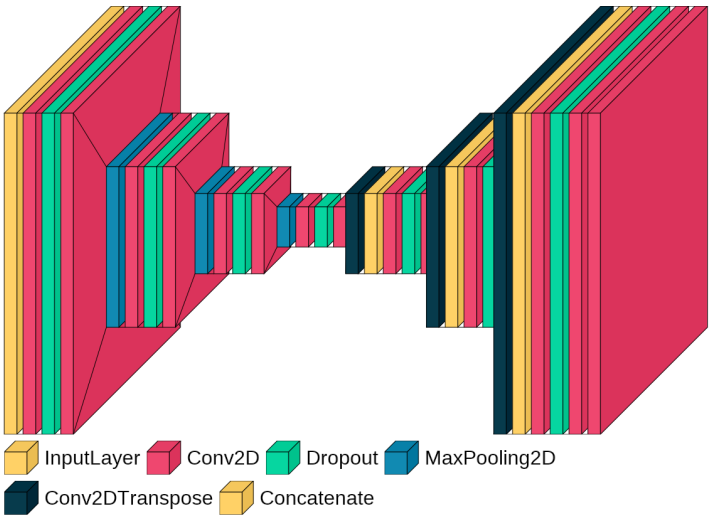


Abbildung 10: Visualisierung des verwendeten Convolutional Neural Network. Vewendet wurden die folgenden Parameter (von links nach rechts der entsprechenden Lage zugeordnet):

- „InputLayer“ Shape: (..., 128, 128, 3)
- „Conv2D“:
- Filter Anzahl: 18, 18, 36, 36, 72, 72, 144, 144, 72, 72, 36, 36, 18, 18, 1
- Kernel Größe: 3 x 3
- Kernel Initialisierungs-Methode: „he_normal“
- Padding: „same“
- Aktivierungsfunktion: ReLU (letzte: Sigmoid)
- „Dropout“ Stärke: 0.1779, 0.1779, 0.2779, 0.2779, 0.2779, 0.1779, 0.1779
- „MaxPooling2D“ Größe: 2 x 2
- „Conv2DTranspose“:
- Filter Anzahl: 72, 36, 18
- Kernel Größe: 2 x 2
- Aktivierungsfunktion: keine
- „Concatenate“: Verkettet Ausgabe vor „MaxPooling2D“ mit Ausgabe nach „Conv2DTranspose“ der gleichen Ebene
- Verlustfunktion: binäre Kreuzentropie
- „Adam“-Lernrate: 0.003795
- „Batch“-Größe beim Training: 128

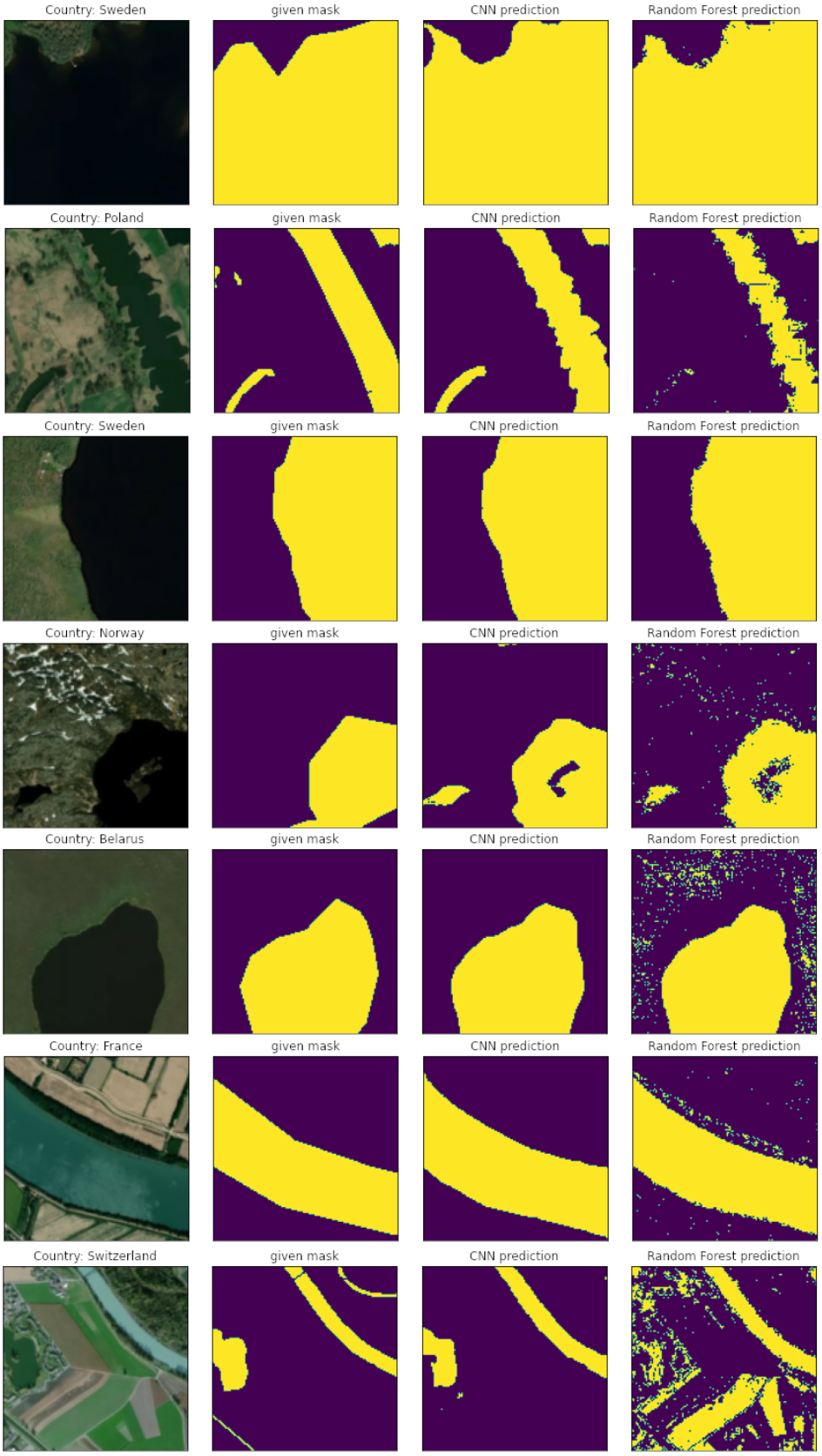


Abbildung 11: Weitere Beispiele zum Vergleich der Hauptmethode zur Alternativmethode.©Mapbox, ©OpenStreetMap

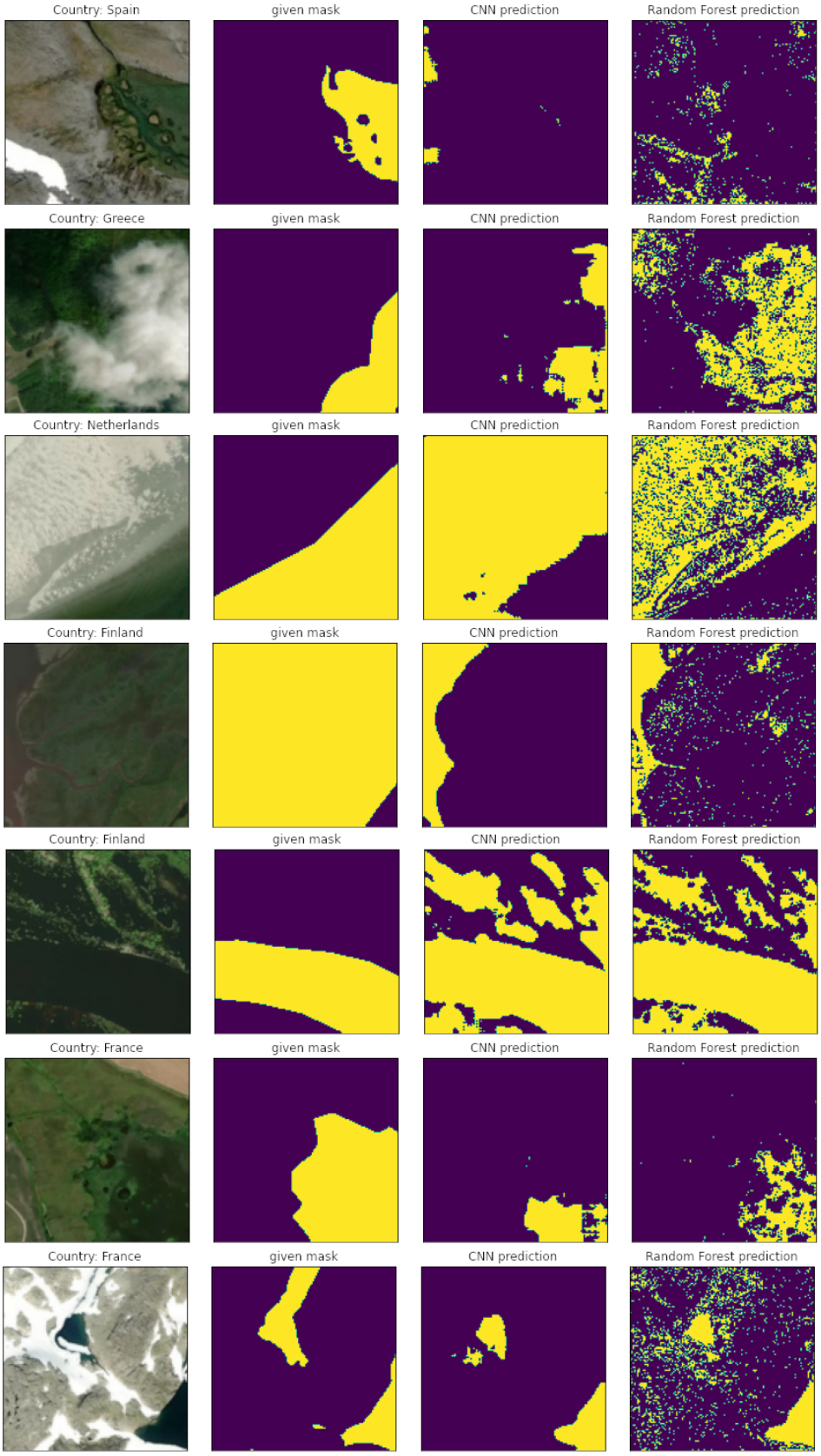


Abbildung 12: Weitere Beispiele zum Vergleich der Hauptmethode zur Alternativmethode, die schleche Ergebnisse lieferten.©Mapbox, ©OpenStreetMap