Machine Learning for Physicists

Wolkenklassifizierung

Noah Biederbeck

31. Juli 2018

Inhaltsverzeichnis

1 Motivation und Fragestellung

Ein langfristiges Projekt ist das Bauen einer autarken Wetterstation basierend auf dem Raspberry Pi[1]. Dies ist ein linuxbasierter Einplatinencomputer (System-On-A-Chip, SoC). Es werden die Temperatur, Luftfeuchtigkeit und Luftdruck von Sensoren gemessen. Langfristig soll die Wetterstation nicht nur aktuelle Wetterdaten aufzeichnen und grafisch aufbereiten, sondern das Wetter auch vorhersagen. Da sich verschiedene Wolkentypen bei unterschiedlichen Witterungsverhältnissen bilden, kam die Idee auf, die aktuelle Wolkenlage als Parameter der Wettervorhersage zu nutzen. Insbesondere die Veränderungen der Wolkenlage lässt eindeutige Rückschlüsse über den Verlauf des Wetters zu.

Die Rechenleistung und der verfügbare Speicher des Raspberry Pi ist gering, sodass es sinnvoll ist, die notwendigen Rechnungen und den Umfang aufgenommener Daten zu beschränken. Aus diesem Grund soll die autarke Wetterstation nur den aktuellen Wolkentyp für die Vorhersage speichern.

Die Wetterstation ist mit einer Kamera bestückt, die regelmäßig Wetterdaten und Himmelfotos aufnimmt. Diese Fotos sollen klassifiziert werden, sodass die Datenspeicherung aus einer Klassenrepräsentation, statt aus einem Foto besteht.

Ein maschinelles Lernverfahren soll genutzt werden, um die Himmelfotos direkt auf dem Raspberry Pi zu klassifizieren. Die Wahl fällt auf ein Neuronales Netz. Neuronal Netze bestehen aus mehreren Schichten, die verschiedene Matrixmultiplikationen verwenden, um zum einen Dimensionen zu reduzieren und zum anderen Eingangsparameter im Bezug auf die zugehörige Klasse zu gewichten. Ein großer Vorteil gegenüber anderen maschinellen Lernverfahren ist, dass das Modell unabhängig von den Trainingsdaten verwendet werden kann (vergleiche kNN, hier muss der gesamte Trainingsdatensatz gespeichert werden), was für die Wettervorhersage sehr sinnvoll ist, da der Speicherplatz begrenzt ist (s. o.). Außerdem besteht so die Möglichkeit, mehrere Wetterstationen zu kombinieren und für die Klassifikation muss nur das Modell weitergegeben werden. Die Matrixmultiplikationen sind der Grund dafür, dass die Auswertungszeit auch mit geringer Rechenleistung kurz bleibt. Ein Neuronales Netz wird mit sogenannten `Convolutional` Schichten die Formen der Wolkenklassen lernen und zusammen

[1] raspberrypi.org

mit den Information der drei Farbkanäle (rot, grün, blau) in `Dense` Schichten gewichten, um eine Klassifikation zu ermöglichen.

Eine Alternativmethode zum Neuronalen Netz ist der Random Forest. Auch hier kann das Modell zum Auswerten ohne die Trainingsdaten weitergegeben werden und ist somit gut geeignet für kleine Systeme wie den Raspberry Pi. Der Random Forest hat weniger Kapazitäten als das Neuronale Netz, da er nur die Information der Farbkanäle zum Trainieren bekommt. Der Random Forest ist ein beliebter Algorithmus für maschinelles Lernen, da er mit wenig Optimierung brauchbare Ergebnisse liefert und seine Trainingszeit sehr gering ist. Ein Random Forest besteht aus einem Ensemble von mehreren binären Entscheidungsbäumen.

2 Auswahl und Beschreibung des Datensatzes

Der Raspberry Pi nimmt regelmäßig Fotos auf. Diese werden von uns in 11 Klassen eingeteilt und gelabelt. Die Verteilung der Klassen ist in Abbildung 1 dargestellt. Dazu haben wir einen `Telegram-Bot`[2] erstellt, der uns und anderen freiwilligen Helfern Fotos geschickt hat, die dann zu labeln sind.

Besonders hervorzuheben sind ‘Bad Pictures’, da auf diesen zum einen gar kein Himmel zu sehen ist, da die Kamera in die falsche Richtung gezeigt hat, zum anderen unbekannte Fehler während der Aufnahme auftraten. Siehe für Beispiele Abbildung 2.

Auf weiteren ‘Bad Pictures’ sind zu viele Regentropfen auf der Kameralinse, sodass eine vernünftige Klassifikation von unserer Seite nicht zuverlässig wäre, was einen unreinen Datensatz zur Folge hätte. Weiterhin nimmt die Wetterstation auch nachts Fotos auf, welche aufgrund des schlechten Lichtsensors in der Kamera nicht gut aufgelöst werden können uns somit großflächig zu dunkel sind, um Wolken zu erkennen.

Fotos bei Nacht und ‘Bad Pictures’ werden nicht mit abgegeben, um den benötigten Speicherplatz für den Datensatz nicht unnötig zu vergrößern. Die beiden Klassen werden auch nicht in das Training der Algorithmen aufgenommen, da die selbstgestellte Aufgabe ist, fotografierte Wolken in Wolkenklassen einzuteilen, nicht zu entscheiden, ob es ein gutes (persönli-

[2] @weatherpi_bot

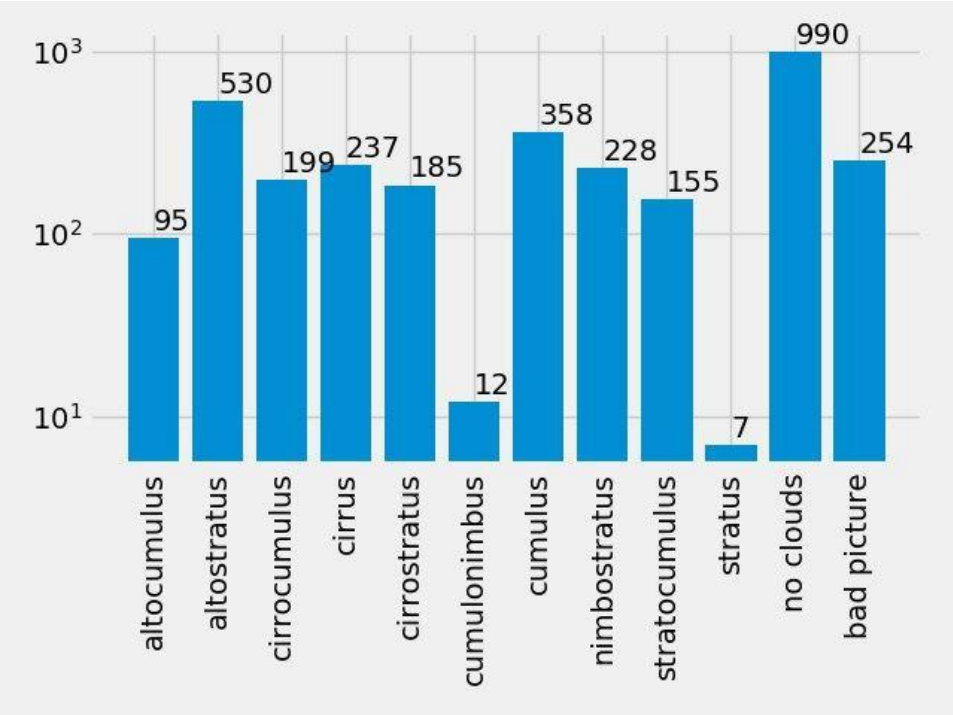
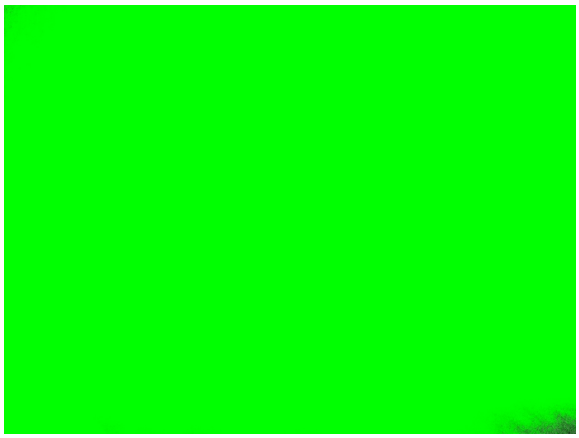


Abbildung 1: Verteilung der Klassen.

che/menschliche Wertung) Foto ist. Auf langfristige Sicht sollte der Anteil von ‘Bad Pictures’ statistisch gering sein.

Der Datensatz besteht ohne ‘Bad Pictures’ und Fotos bei Nacht aus 3250 Fotos mit 1024×768 Pixeln mit je 3 Farbkanälen und belegt unkomprimiert 1,5 GB Speicherplatz. Er ist aufgeteilt in 11 Wolkenklassen, von denen 2 aus weniger als 12 Fotos bestehen.

Der Datensatz wird freigegeben unter der MIT-Lizenz.



(a) Unbekannter Fehler während der Aufnahme.

(b) Regentropfen auf der Kameralinse erschweren das Labeln.

(c) Beispiel für „falsche Richtung“.

Abbildung 2: Beispiele für ‘Bad Pictures’.

3 Motivation und Darstellung des Lösungsansatzes

Im Folgenden wird erklärt, wie die aufgenommenen Fotos vorverarbeitet werden und wie sich die Architekturen und das Training der maschinellen Lernverfahren auf die gewählten Leistungsmerkmale auswirken.

3.1 Vorverarbeitung

Zuerst werden Fotos bei Nacht aus dem Datensatz entfernt. Die erste Idee war, die Fotos anhand der Aufnahmezeit zu filtern. Dies wurde schnell verworfen, da bewusst war, dass bei einer langfristigen Nutzung der Wetterstation die Zeiten der Sonnenauf- und -untergänge variieren. Die Fotos werden stattdessen bei einer mittleren Pixelhelligkeit unter 70 verworfen. Dies hat den Vorteil, dass keine Rücksicht auf den Standort und die Systemzeiteinstellung verschiedener Wetterstationen Rücksicht genommen werden muss.

Die Aufnahme ist so programmiert, dass die Fotos noch vor dem Abspeichern den Helligkeitsfilter durchlaufen, sodass dunkle Fotos nicht gespeichert werden.

Anschließend werden Schnitte auf den Farbkanälen der einzelnen Fotos angewendet. Dies geschieht aus zwei Gründen, die in der dargestellten Reihenfolge während der Arbeit aufkamen:

- Entfernung von fotografierten Objekten, die nicht zum Himmel gehören. Hierzu zählen insbesondere Bäume und Hauswände, die sich im Sichtfeld der Kamera befinden. Diese Objekte sollen mit einem allgemein anwendbaren Algorithmus, anstelle eines harten Schnittes auf entsprechenden Bildregionen, entfernt werden, damit verschiedene Wetterstationen denselben Code verwenden können.
- Entfernung von Farbwerten wie grün und rot, die im Farbraum von den dominanten Himmelfarben blau, weiß und grau weit entfernt sind, damit die Parameter des Random Forest die Himmelfarben und nicht Umgebungsfarben sind.

Es wird hierzu ein Schnitt der Form einer im Farbraum rotierten Parabel mit

$$b > (c - x_0)^2 + x_1, \tag{1}$$

mit b den Blauwerten der Pixel und c der zu schneidenden Farbe (grün, rot), auf den Farbkanälen

angewendet. Die Parameter x_0 und x_1 werden mit dem Helligkeitswert der Pixel verschoben. Die Pixel, die geschnitten werden sollen, werden auf den RGB Wert $(0,0,0)$, also schwarz gesetzt.

Es werden die einzelnen Kanäle im Bereich $[1,255]$ in 30 Bins histogrammiert und normiert.

In Abbildung 3 ist dieser Farbschnitt und die Histogramme für ein Beispielfoto dargestellt.

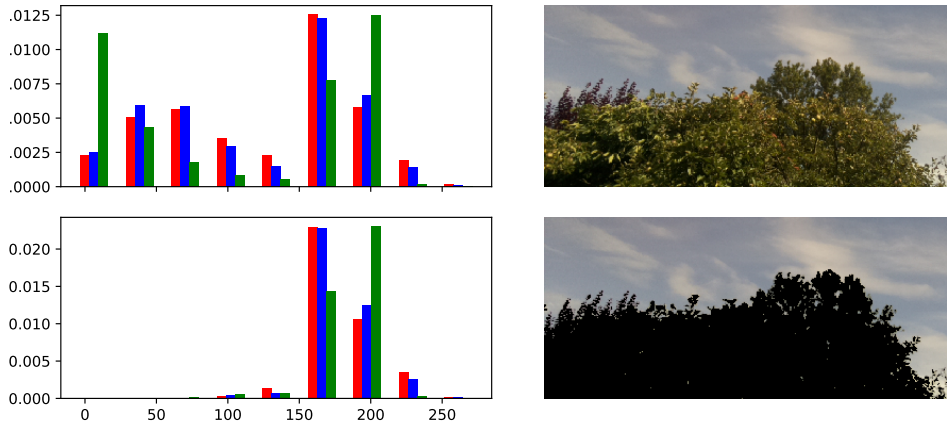


Abbildung 3: Histogramm und Originalfoto mit und ohne Farbschnitt, dargestellt sind nur 10 Bins.

Der Farbwert 0 wird nicht in das Histogramm aufgenommen, da sonst die durch die Farbschnitte entfernten Objekte in die Trainingsdaten aufgenommen werden würden.

Im letzten Schritt der Vorverarbeitung wird der Datensatz aufgeteilt in Trainings- und Testdatensatz, die separat in Unterordner gespeichert und dort in die jeweiligen aufgeteilt Klassen werden. Dies ist notwendig, damit im Training des Neuronalen Netzes die Funktion `flow_from_directory` verwendet werden kann, die Batch-weise Fotos aus den Ordnern lädt, um den Arbeitsspeicher nicht zu überlasten.

Für das Training werden nur 90 bis 150 Bilder pro Klasse verwendet. Das bedeutet, dass die Klassen ‘cumulonimbus’ und ‘stratus’ nicht klassifiziert werden können. Die Ausgeglichenheit des Datensatzes ermöglicht jedoch ein von der relativen Häufigkeit der Klassen unabhängiges Training.

3.2 Neuronales Netz

Ein Neuronales Netz wird verwendet, da es in der Lage ist, mittels `Convolutional` Schichten Formen auf Bildern zu erkennen. Die Auswertung besteht aus Matrixmultiplikationen, die einfach zu rechnen sind. Das Neuronale Netz wird mit dem Python Paket *keras*[3] implementiert. Es wird die Klasse `keras.models.Sequential` verwendet.

Die Eingangsparameter des Neuronalen Netzes sind die vollständigen Fotos, also $1024 \times 768 \times 3$ Parameter. Die erste Schicht ist eine `AveragePooling2D` Schicht, die dazu dient, die Dimension schnell zu reduzieren. Es werden 3 Abfolgen von `Convolutional` Schichten und `MaxPooling` Schichten verwendet. Die Schichten werden zur Dimensionsreduktion verwendet. Zusätzlich dazu ist die `Convolutional` Schicht geeignet, mittels Faltung Kanten zu finden und somit fähig, die Form der Wolken zu lernen. Im Anschluss folgt eine `Flatten` Schicht, die die $(n \times N)$-Dimensionalität auf eine $(n \cdot N \times 1)$-Dimension reduziert. Dies ist notwendig, um im Anschluss klassische `Dense` Schichten zu verwenden.

Es folgen `Dense` Schichten mit 32, 32 und 9 Neuronen, jeweils gepaart mit `GaussianNoise` Schichten und `Dropout` Schichten, zur Regularisierung. Die `Dropout` Schichten werden genutzt, um die Gewichte in einer Größenordnung zu halten, die `GaussianNoise` Schichten, um die Gewichte klein zu halten. Beides ist in der Matrixmultiplikation von Vorteil, da so die genaue numerische Darstellung von Fließkommazahlen im Rechner möglich ist.

Die Aktivierungsfunktionen sind `ReLU` in den `Dense` Schichten und `Sigmoid` in der Output-Schicht, um die Vorhersagen zu normieren. Als Lossfunktion wird `logcosh` verwendet, da dieser fehlerhafte Vorhersagen linear bestraft.

3.3 Random Forest

Ein Random Forest wird verwendet, da er mit geringem Trainingsaufwand sehr brauchbare Ergebnisse erzielt. Das Modell ist klein und leicht rechenbar. Der Random Forest wird mit dem Python Paket *scikit-learn*[4] implementiert. Es wird die Klasse `sklearn.ensemble.RandomForestClassifier` verwendet.

[3] keras.io

[4] scikit-learn.org

Die Eingangsparameter des Random Forest sind die histogrammierten Farbkanäle der Fotos, auf denen die Farbschnitte angewendet wurden. Die Trainingsdaten haben die Dimension $(N \times 90)$.

Der einzige wichtige Hyperparameter ist hier `n_estimators`. Er entscheidet, wie viele Bäume im Random Forest verwendet werden. Es wird `n_estimators` $= 300$ gewählt, da durch eine hohe Anzahl an Bäumen Overfitting verhindert wird.

4 Darstellung und Interpretation der Ergebnisse

Die nachfolgend dargestellten Untersuchungen des Datensatzes werden mit Random Forest durchgeführt, da dieser um Größenordnungen schneller trainiert.

4.1 Nichtübereinstimmung der Label auf dem Datensatz

Bei der Einteilung des Datensatzes in die Unterordner der jeweiligen Klassen des Trainings- und Testdatensatzes ist aufgefallen, dass die zugeordneten Label teilweise nicht mit den Klassen übereingestimmt haben. Bei der Inspektion des Codes für das Labeln ist klar geworden, dass sich parallele Threads beim Labeln mehrerer Personen gleichzeitig überschrieben haben und somit falsche Label zu den Fotos zugeordnet worden sind.

Um die falsch gelabelten Fotos zu korrigieren, wird ein simpler Random Forest auf einem kleinen Teil des fehlerhaften Datensatz trainiert und auf dem gesamten Datensatz ausgewertet. Bei Klassifizierungen, die nicht mit dem von uns gesetzten Label übereinstimmten, haben wir mit einer binären Entscheidung („stimmt“ / „stimmt nicht“) das von uns gesetzte Label bestätigt oder verworfen. Dazu wurde der Code zum Labeln und der Telegram-Bot erweitert. Die verworfenen Fotos mussten neu gelabelt werden.

Dieser Vorgang wurde mehrere Male wiederholt, bis ein fehlerfrei gelabelter Datensatz entstand.

Ein Random Forest ist zusätzlich zu den oben genannten Argumenten für diese Aufgabe geeignet, da dieser robuster als ein Neuronales Netz gegen falsche Label im Training ist.

Im Anschluss können sowohl ein finaler und auf reinem Datensatz trainierter Random Forest

sowie ein Neuornales Netz trainiert werden.

4.2 Neuronales Netz

Das Neuronale Netz wird 200 Epochen trainiert. Als Metrik wird `categorial_accuracy` verwendet. Dies ist die Genauigkeit bei der Klassifizierung einer einzelnen Klasse, gemittelt über alle Klassen.

Die Genauigkeit und der Loss werden auf den Trainings- und Testdaten gegen die Epoche in Abbildung 4 dargestellt. Es ist zu erkennen, dass die Werte in Sättigungswerte laufen, die für den Validierungsdatensatz (gepunktete Linie) bei

$$ACC = 0{,}65 \tag{2}$$

$$LOSS = 0{,}35 \tag{3}$$

liegen.

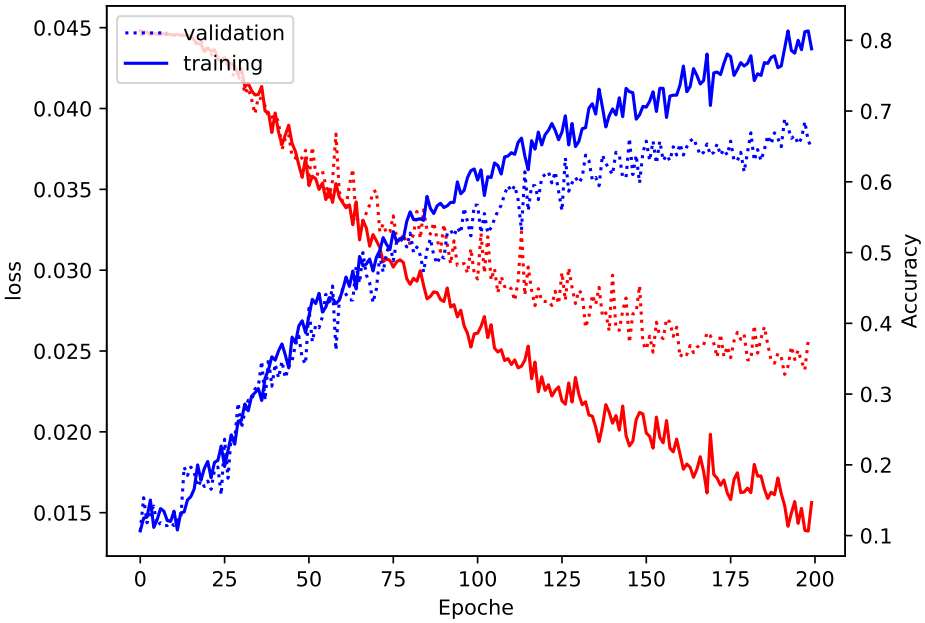


Abbildung 4: Trainings- und Validierungsloss und -genauigkeit.

Die Confusion Matrix in Abbildung 5a zeigt, welche Klassen besser und welche schlechter

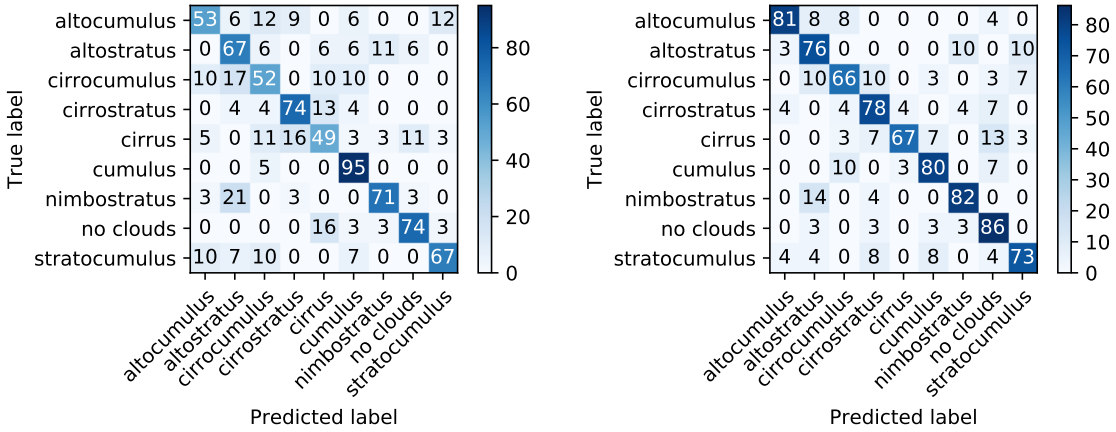
vorhergesagt werden. Insbesondere die Klasse ‘cumulus’ wird mit 95% Genauigkeit vorhergesagt. Dies liegt an der deutlich ausgeprägten Form und und den harten Unterschieden in den Farben. Ein Beispiel für eine Cumuluswolke ist im Anhang in Abbildung 7 zu finden. Die Klasse ‘cirrus’ wird schlecht vorhergesagt. Gründe hierfür sind geringe Farbunterschiede und diffuse Formen. Ein Beispiel für eine Cirruswolke ist im Anhang in Abbildung 8 zu finden.

4.3 Random Forest

Der finale Random Forest, der die Alternativmethode zum Neuronalen Netz darstellt, erreicht eine Genauigkeit auf dem Testdatensatz von 74%. An der Confusion Matrix in Abbildung 5b ist eindeutig zu erkennen, welche Wolkenklassen gut, und welche schlecht voneinander getrennt werden können.

Eindeutig zu erkennen ist, dass die Klassen ‘nimbostratus’ und ‘no clouds’ gut vom Random Forest vorhergesagt werden kann, während die Klassen ‘cirrus’ und ‘cirrocumulus’ schlecht klassifiziert werden können. Dies liegt im ersten Beispiel an der hohen Homogenität der Farbe über das gesamte Foto. Dass insbesondere ‘cirrus’ als ‘no clouds’ klassifiziert wird, liegt an der diffusen Form der Cirruswolke und den deswegen geringen Farbunterschieden der beiden Fotos. Siehe auch hier Abbildung 8.

Die Auswertungszeiten der Modelle sind in Abbildung 6 dargestellt.



(a) Confusion Matrix für das Neuronale Netz.

(b) Confusion Matrix für den Random Forest.

Abbildung 5: Confusion Matrizen der Modelle.

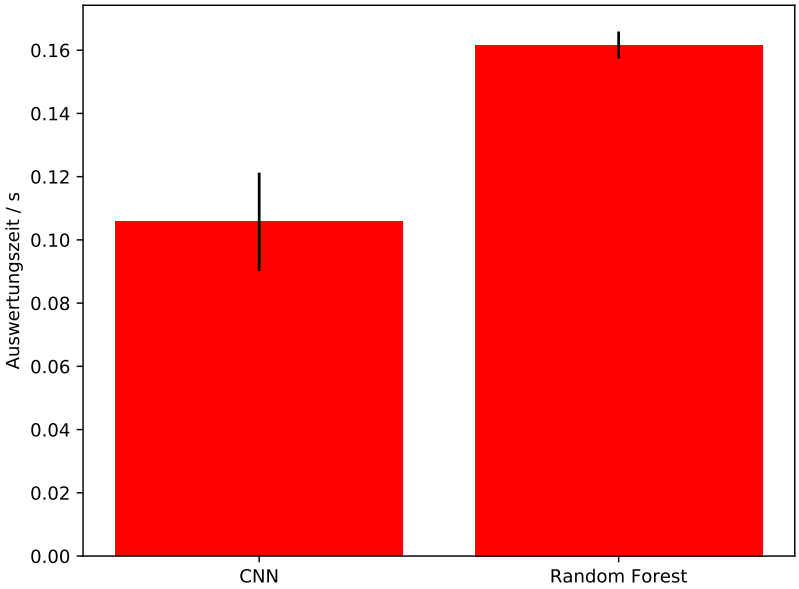


Abbildung 6: Vergleich der Auswertungszeiten von Neuronalem Netz und Random Forest.

5 Zusammenfassung

Ein erstes Ergebnis dieser Arbeit ist die Bestätigung, dass ein Random Forest robust gegen falsche Label im Datensatz ist. Dies liegt an der Mittelung über viele Entscheidungsbäume. Der Random Forest erreicht eine höhere Leistung, als das Neuronale Netz, bezogen auf die Problemstellung.

Tabelle 1: Vergleich der relevanten Metriken.

Merkmal	Neuronales Netz	Random Forest
Genauigkeit / %	64	74
Auswertungszeit / s	0.11	0.16

Jedoch ist das Neuronale Netz in der Auswertung schneller, als der Random Forest (s. Tabelle 1). Es muss berücksichtigt werden, dass der Random Forest einen Trainingsdatensatz einer viel geringeren Dimension hat.

Aus diesen Gründen ist die Alternativmethode für die Wolkenklassifikation besser geeignet, als

die gewählte Methode des maschinellen Lernens.

Es hat sich herausgestellt, dass die Problemstellung mit einem recht simplen Algorithmus besser beantwortet werden kann, als mit einem komplizierteren. Dies ist jedoch nur möglich, da die Vorverarbeitungsschritte gut überlegt und implementiert sind. Der Vorteil der Implementierung der Vorverarbeitung ist, dass jedes mögliche Foto gefiltert werden kann, und sie nicht auf die Bildgröße 1024×768 beschränkt ist. Außerdem kann ein anderer Farbraum geschnitten werden, wenn eine gute Schnittfunktion gefunden wurde.

Die durchschnittliche Genauigkeit von 74% bei einer Klassifikation in 9 Klassen ist ein sehr guter Wert. An der Confusion Matrix wird deutlich, welche Klassen besser und welche schlechter vorhergesagt werden können. Diese Klassen unterscheiden sich in den Daten sehr gering, sodass zusätzliche Vorverarbeitungscchritte sinnvoll sind. Ein mehrstufiges System, das zum Beispiel erst Wolken von ‘no clouds’ trennt und im Anschluss die Cirrus-Wolkentypen einzeln klassifiziert, kann hier von Vorteil sein.

Anhang

Um den Code auszuführen, sollte eine virtuelle Umgebung für Python verwendet werden. Mit iner aktuellen Installation von `conda` ist die möglich. Im Ordner `weatherpi` wird `pip install -e .` und `pip install -r requirements.txt` (in der Reihenfolge) gerufen, um die notwendigen Pakete zu installieren. Im Ordner `weatherpi/predictions/pipeline/` wird mit `python pipeline.py` die gesamte Analyse durchgeführt. Die Daten liegen in `weatherpi/predicitons/pipeline/data` und die Label in der Datei `weatherpi/predictions/pipeline/label.pkl`.

Im Folgenden sind die Kommandos erneut aufgelistet, auszuführen im Order `weatherpi`.

```
conda create -n bdrbcksckl python=3.6 pip
conda activate bdrbcksckl
pip install -e .
pip install -r requirements.txt
cd predictions/pipeline
python pipeline.py
```

Abbildung 7: Beispiel für eine Cumuluswolke. Gut zu erkennen sind Form und Farbunterschiede. Dies sind die Gründe für die gute Klassifikation des Neuronalen Netzes.

Abbildung 8: Beispiel für eine Cirruswolke. Gut zu erkennen sind die diffuse Form und geringe Farbunterschiede. Dies sind die Gründe für die schlechte Klassifikation des Neuronalen Netzes.