Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

Bericht Projekt BDP 2018

Wolkenkamera

von

Marcel Steger, Jan Zickermann

Fakultät für Mathematik, Informatik und Naturwissenschaften
Fachbereich Informatik
Arbeitsbereich Wissenschaftliches Rechnen

Studiengang: Wirtschaftsinformatik	Informatik
Matrikelnummern: 6592795	6824246

Betreuer: Tobias Finn, Dr. Julian Kunkel

Hamburg, 2018-03-31

Abstract

In diesem Projekt wird von Bildern einer Wolkenkamera die Wolkenhöhe abgeleitet. Hierzu wird ein *convolutional neural network* trainiert, das eine Höhenvorhersage berechnet. Es werden verschiedene Ansätze für die Höhenvorhersage beleuchtet und die Implementation eines Modells zur Höhenklassifikation beschrieben.

Contents

1 Einleitung

Die Ansammlung und spätere Verarbeitung und Analyse von großen Datenmengen ist längst zum Alltag geworden. Ob es sich um einen Skandal bei der Auswertung von persönlichen Daten oder nur um einen Kaufvorschlag beim Online-Shopping handelt. *Big Data* ist heutzutage allgegenwärtig.

Eine sehr große Datenansammlung entsteht unter anderem durch eine stationäre Wolkenkamera, welche alle 30 Sekunden eine Aufnahme macht. Die Kamera gehört zu den meteorologischen Geräten vom Wettermast Hamburg[1] und wird unter anderem für die Bestimmung des Bewölkungsgrads verwendet. Diese Aufnahme bildet den gesamten Himmel und das darauf zu sehende Wolkenbild in einer Fisheye-Ansicht ab. Aus den hierbei entstehenden Datenmengen (2880 Bilder pro Tag) können durch *machine learning* nützliche Informationen extrahiert werden. Zusätzlich stehen neben den Bilddaten weitere, für das *machine learning* hilfreiche, Datenmengen zur Verfügung. Zu diesen Messdaten gehören Regendetektion, die Regenmenge, Sonnendetektion und die Sonneneinstrahlung. Das Ziel bei der Verarbeitung der Aufnahmen ist die Ermittlung der Höhe der abgebildeten Wolken. Diese sollen durch Training eines neuronalen Netzes mit Hilfe der weiteren Messdaten festgestellt werden.

[1] Wettermast Hamburg: `https://wettermast.uni-hamburg.de/frame.php?doc=Messanlage.htm`

2 Aufgabenstellung

Basierend auf den Bildern einer Wolkenkamera soll eine Vorhersage über die Wolkenhöhe getroffen werden. Hierfür tragen die im Bild sichtbaren Wolkenstrukturen Informationen über die Wolkenhöhe. Zudem geben Bedeckungsgrad, Regen und Sonneneinstrahlung Hinweise auf die Wolkenhöhe des Bildes. Um diese Eigenschaften aus Strukturen des Bildes abzuleiten, werden *convolutional neural networks (ConvNets)* verwendet. Hierzu wird ein Trainingsdatensatz aus den verfügbaren Messdaten erstellt und dazu genutzt ein *ConvNet* zu trainiert. Das *ConNet* soll möglichst genaue Höhenvorhersagen treffen und über den Trainingsdatensatz hinaus generalisierbar für weitere Wolkenkamerabilder sein.

3 Realisierung

3.1 Design

3.1.1 Bildprojektion

Die Wolkenbilder sind durch eine Fischaugenlinse so verzerrt, dass Größenproportionen nicht einheitlich sind. Damit Bildmuster unabhängig von ihrer Position im Bild erkannt werden können, muss das Bild auf die Wolkenebene projiziert werden. In [Lange(2016)] wird hierzu beschrieben, wie zu jedem Bildpunkt des Zielbilds der entsprechende Punkt des Ursprungsbilds zu berechnen ist. Zudem wird die Hemisphärenkoordinate ϵ_{ideal}, die für wahre Projektionen der Fischaugenlinse abweicht, mit dem Polynom P korrigiert. Somit wird die Genauigkeit der Projektion mit $\epsilon_{\text{wahr}} = P(\epsilon_{\text{ideal}})$ erhöht.

3.1.2 Messdatenauswahl

Die Ableitung einer Wolkenhöhe aus einer Bildaufnahme ist nur sinnvoll, wenn das Bild eine ausreichende Helligkeit besitzt, um Wolken klar erkennen zu können. Nach [Lange(2016)] ist bereits in der Dämmerung die Wolkenerkennung durch eine starke Blautönung erschwert. Um keine Bilder dieser sogenannten blauen Stunde zu verwenden, werden nur Bilder mit einem Sonnenwinkel von $> 7,5°$ über dem Horizont betrachtet.

Die Wolkenkamera nimmt alle 30 Sekunden ein Bild auf, während der Ceilometer nur alle 60 Sekunden eine Höhenmessung vornimmt. Da eine Interpolation der Höhenwerte keine realistische Höhe ergibt, wird der zum Bildzeitstempel zeitlich nächste Höhenwert gewählt. Regen- und Sonnendaten werden ebenfalls im 60 Sekundentakt gemessen [Lange(2018)]. Die Regenmenge in mm (RR) nimmt selten Werte > 0 an. Um eine wertvolle Zuordnung zum Bild zu erhalten, wird die maximale Regenmenge aus Messwerten mit Abständen in dem Intervall $[-10, 10]$ Minuten dem Bildzeitstempel zugeordnet. Für die Detektion $\in \{0, 1\}$ von Regen (RD) wird der häufigste Wert im Intervall $[-30, 0]$ Minuten ausgewählt. Die kurzwellige Sonneneinstrahlung W/m^2 (G) nimmt kontinuierliche Werte an, sodass die Bildung eines Mittelwerts sinnvolle Werte ergibt. Es wird dem Bildzeitstempel der Mittelwert im Intervall von $[-5, 5]$ zugeordnet. Für die Sonnendetektion $\in \{0, 1\}$ (GSD) wird der häufigste Wert in $[-5, 5]$ Minuten verwendet. Der verwendete Datensatz besteht somit insgesamt aus Höhenwerten, entzerrten Wolkenbildern, Regen- und Sonnendaten jeweils zugeordnet zu einem Bildzeitstempel.

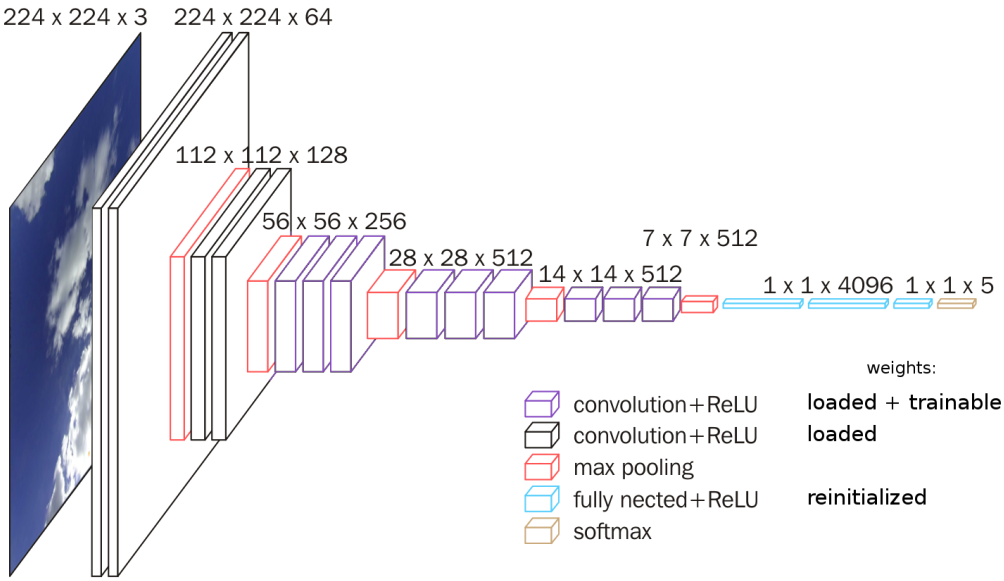

Figure 3.1: VGG16 Architektur. In Anlehnung an: [Frossard(2016)]

3.1.3 Convolutional Neural Network

ConvNet Architekturen eignen sich besonders für Bilderkennungsprobleme, da sie weniger Gewichtungen benötigen als gewöhnliche Neuronale Netze und mit *ImageNet* Klassifikationen bereits ihre Leistungsfähigkeit gezeigt haben [Alex Krizhevsky and Hinton(2012)]. Eine dieser leistungsfähigen *ConNet* Architekturen ist *VGG16* [Karen Simonyan(2014)]. Da dieses Netz viele Trainingsbilder benötigt und nicht genügend Wolkenbilder direkt zur Verfügung stehen, werden Gewichtungen ausgehend eines zuvor trainierten Modells vor dem Training mit den Wolkendaten geladen. Die Gewichtungen von *layer* 1 bis 4 des *VGG16* werden jedoch nicht verändert, da diese ersten *layer* voraussichtlich allgemeine und wichtige Bildverarbeitungen darstellen, die auch für die Wolkenerkennung zutreffend sind. Dahingegen werden die Gewichtungen der *fully connected (FC) layer* reinitialisiert, da hier Gewichtungen spezifisch für die neue Wolkenerkennungsaufgabe erlernt werden sollen. Die Abbildung 3.1 stellt die verwendete *VGG16* Architektur dar.

3.1.4 Ansätze für die Höhenvorhersage

Der erste Ansatz ist einen reellen Wert als Höhenvorhersage zu bestimmen. Sei t_i die Wolkenhöhe des Bildes i und y_i die zugehörige Höhenvorhersage. Es wäre ein Modell für eine Höhenregression denkbar mit absoluter Differenz $|y_i - t_i|$ als *loss* Funktion. Um hingegen den *loss* in niedrigen Wolkenhöhen zu verstärken und in hohen Höhen zu verringern, kann die *loss* Funktion als $|\log(y_i + 1) - \log(t_i + 1)|$ gewählt werden. Besitzt jedoch ein Bild i keine Wolkenhöhe d.h. $t_i = \infty$, so kann kein reeller *loss* berechnet werden. Eine Lösung wäre alle $t_i = \infty$ auf einen reellen Wert s abzubilden. Beispielsweise $s := 2 \cdot \max_{1 \leq i \leq n} h_i$ oder $s := 0$. Alternativ könnte die *loss* Funktion so angepasst werden,

height_class	range (m)
0	≤ 360
1	≤ 840
2	≤ 1530
3	> 1530
4	∞

Figure 3.2: Ausbalancierte Höhenklassen für *train/validation* Datensatz

dass für $t_i = \infty$ eine Wahrscheinlichkeit u_i, wie in (3.1), einbezogen wird.

$$l(y, u, t) = \begin{cases} |y - t| + \log(\frac{1}{1-u}) & \text{falls } t < \infty \\ \log(\frac{1}{u}) & \text{sonst} \end{cases} \tag{3.1}$$

In (3.1) werden so zwei Klassen unterschieden: $t = \infty$ und $t < \infty$. Nun ist ein Ansatz mit weiteren Klassen denkbar, sodass neben der Höhenregression auch Höhenklassen berechnet werden. Für die Klassifikation würde *cross entropy* als *loss* Funktion dienen. Ferner können weitere Klassifikationen und Regressionen mit Regenmenge (RR), Regendetektion (RD), Sonneneinstrahlung (G) und Sonnendetektion (GSD) dem Modell hinzugefügt werden. Jedoch ist dies nur sinnvoll, sofern die Regen- und Sonnenwerte dem Modell helfen die Wolkenhöhe zu bestimmen. Zudem müssen für diese verschiedenen unterstützenden *tasks* eine Gewichtung in der *loss* Funktion angewendet werden, sodass der gesamte *loss* die Qualität des Modells widerspiegelt.

Der letzte Ansatz ist eine reine Höhenklassifikation. Die Höhenklassen werden hierbei so gewählt, dass im verwendeten Datensatz die Klasse $t = \infty$ mit möglichst gleicher Häufigkeit, wie die anderen Klassen vorkommt. Da $5055/24693 \approx \frac{1}{5}$ des Datensatzes aus Wolkenhöhen mit $t = \infty$ besteht, werden 4 weitere, auf reellen Höhen ausgewogene, Höhenklassen unterschieden, um insgesamt 5 ausgewogene Höhenklassen für den gesamten Datensatz zu erhalten. Die ausbalancierten Höhenklassen erlauben dem Modell, ohne eine Gewichtung der Klassen, die Bildklassifikation zu erlernen und eine einfache Interpretation des Lernerfolgs ist möglich.

3.2 Implementierung

3.2.1 Preprocessing

Die Wolkenbilder eines Tages liegen jeweils in einem *zip* Archiv vor. Der Tag des Archivs ist im Dateinamen enthalten und in den Namen der Bilder im Archiv sind die Zeitstempel der Bilder enthalten. Diesen Zeitstempeln werden Wolkenhöhen aus den Ceilometer-Daten zugeordnet. Vorher werden Zeitstempel mit zu niedrigen Sonnenhöhe vor weiterer Verarbeitung herausgefiltert. Für die Bestimmung des Sonnenwinkels wird das Paket *pysolar* verwendet. *pysolar* berechnet die Sonnenhöhe aus dem Längen- und Breitengrad der Kamera und dem Zeitstempel des Bildes. Die Ceilometer-Daten

liegen in monatlichen Verzeichnissen in täglichen *csv* Dateien vor. Ein Beschreibung der Kürzel kann in [Lange(2015)] unter *CT25* gefunden werden. Die unterste Wolkenbasis m ($CTWBU$) wird als Messwert für die Wolkenhöhe verwendet. Wenn keine Wolkenhöhe vorliegt, wird ∞ als Messwert angenommen. Die Höhenmessung mit dem Zeitstempel, der dem Bildzeitstempel am nächsten ist, wird ausgewählt. Sollte kein passender Zeitstempel mit einer maximalen Abweichung zum Bildzeitstempel von 90 Sekunden gefunden werden, so wird das zugehörige Bild nicht in den Datensatz aufgenommen.

Nachdem einem Bildzeitstempel eine Höhe zugeordnet wurde, werden noch Regen- und Sonnendaten hinzugefügt. Hierfür wird, wie in 3.1.2 beschrieben, für jeden Zeitstempel ein Wert für jeweils *RR*, *RD*, *GSD* und *G* bestimmt. Alle Datenpunkte werden in eine zufällige Reihenfolge gebracht und $\frac{4}{5}$ werden dem *train* Datensatz und $\frac{1}{5}$ dem *validation* Datensatz zugeordnet. Anschließend wird für jeden Zeitstempel das zugehörige Bild aus dem Archiv gelesen und nach 3.1.1 auf die Wolkenebene projiziert. Die Bilder werden auf die Größe 224×224 Pixel für die Verwendung mit *VGG16* verkleinert. Alle Datenpunkte werden als *Example* mit Bild, Höhe, Regen- und Sonnendaten als *Features* in eine *tfrecords* Datei geschrieben. Das *tfrecords* Format ist für das Training des Modells mit *TensorFlow* vorgesehen. Das *label* für die Höhenklasse wird erst nach dem Auslesen der *Examples* mit der Höhe berechnet.

3.2.2 Training des Modells

In Anlehnung an [Frossard(2016)] wurde das *VGG16* Modell in *TensorFlow* implementiert. In [Frossard(2016)] wurde das *Caffe* Modell mit einem Tool[1] in ein *TensorFlow* Modell konvertiert. Zudem wurden für *TensorFlow* konvertierte *pre-trained* Gewichtungen bereitgestellt, welche hohe Performance für *ImageNet* erreichen. Da die Gewichtungen mit 14 Millionen *ImageNet* Bildern erlernt wurden und mit den wenigen Wolkenbilder nur eine Spezialisierung des Modells erlernt wird, werden für die Wolkenhöhenklassifikation die ersten 4 *layer* nicht verändert. Zudem wird der gleiche *RGB* Farbmittelwert $[123.68, 116.779, 103.939]$ von den Bildern subtrahiert. Für die Höhenklassifikation berechnet der *softmax classifier* nur Wahrscheinlichkeiten für 5 Höhenklassen, anstatt 1000 Klassen bei *ImageNet*.

Das Modell nutzt einen *AdamOptimizer*, der *cross entropy* minimiert. Es wurden die *default* Parameter von *TensorFlow*, bis auf die *learning rate* $= 10^{-4}$, verwendet. Die Klassifikation wird mit *batch-size* $= 10$ über mehrere Epochen mit dem $\frac{4}{5}$ *train* Datensatz trainiert. Periodisch werden mit dem *validation* Datensatz *tensorboard summaries* berechnet, um die Qualität des Modells einzuschätzen und *overfitting* zu erkennen. Um den Trainingsfortschritt des Modells zu sichern, werden nach jeder Epoche *checkpoints* angelegt. So kann das Modell zu einem späteren Zeitpunkt weiter trainiert oder getestet werden.

Um *overfitting* des Modells bei wenigen Trainingsdaten zu vermeiden und ein besser generalisierendes Modell zu erhalten, sind *dropout* und *augmentation* der Bilder geeignet [Alex Krizhevsky and Hinton(2012)]. Für die letzten beiden *fully connected layers* wer-

[1] Caffe to TensorFlow: `https://github.com/ethereon/caffe-tensorflow`

height class (m)	correct/samples	accuracy	$H(t, y)$: correct/all
≤ 360	842/992	0.85	0.13/0.52
≤ 840	690/960	0.72	0.25/0.77
≤ 1530	607/982	0.62	0.30/0.93
> 1530	818/989	0.83	0.21/0.56
∞	808/1016	0.80	0.13/0.58
all	3765/4939	0.76	0.20/0.67

Figure 3.3: *Accuracy* und *cross entropy* der Höhenklassen für *validation* Datensatz nach 20 Epochen

den 0.5 *dropout* eingesetzt. Zudem wurden die Bilder zufällig auf der Horizontalen oder Vertikalen gespiegelt, um die Trainingsdaten zu erweitern.

3.3 Leistungsanalyse

Nach 20 Epochen Training mit *augmentation* und *dropout* aktiv, hat das Modell nicht mehr den *loss* des *validation* Datensatzes verringert und *overfitting* hat sich durch ein dennoch langsam sinkenden *loss* des *train* Datensatzes gezeigt. Die Performance des Modells kann in der Tabelle 3.3 nachvollzogen werden. Die Ergebnisse wurden mit dem *validation* Datensatz berechnet, welcher nicht für *backpropagation* verwendet wurde. Die niedrigste Höhenklasse besitzt die höchste Genauigkeit mit 0.85. In dieser Höhenklasse befinden sich die Wolken auf einer niedrigen Höhe, die eng mit Regen und einem hohen Bedeckungsgrad zusammenhängt. Dies resultiert in gleichartigen Bildern, was die hohe Genauigkeit für diese Klasse erklären kann. Auf ähnliche Weise können die hohen Genauigkeiten für die Klassen > 1530 und ∞ interpretiert werden, da wolkenarme und wolkenlose Bilder geringere Variationen besitzen. Die geringeren Genauigkeiten der Klassen ≤ 840 und ≤ 1530 können mit variationsreichen Wolkenbildern zusammenhängen. Hingegen könnte die Ungenauigkeit auch in unklaren Unterscheidungsmerkmalen der Klassen begründet sein. Jedoch ist dies aufgrund der Größe der Klassen unwahrscheinlicher.

Die durchschnittliche *cross entropy* $H(t, y)$ berücksichtigt die vorhergesagten Wahrscheinlichkeiten für falsche Klassen, um den Fehler der getroffenen Vorhersage zu bestimmen. Für den Wert *all* wird die *cross entropy* aller Vorhersagen und für *correct* nur von korrekten Vorhersagen berücksichtigt. So erhöht sich *correct* $H(t, y)$, wenn die korrekte Klasse die höchste Wahrscheinlichkeit besitzt, jedoch für die anderen Klassen höhere Wahrscheinlichkeiten berechnet werden. So könnte der *correct* Wert als Unsicherheit bei einer korrekten Klassifikation interpretiert werden. Die korrekten Klassifikationen für die Klassen ≤ 360 und ∞ sind mit 0.13 am sichersten, Klasse ≤ 1530 mit 0.30 am unsichersten. Dies kann durch die klare Unterscheidbarkeit eines klaren Himmels bei ∞ zu bewölkten Bildern erklärt werden und die besonders starke Bewölkung bei ≤ 360. Diese Extrema der Wolkenbedeckung sind mit größerer Sicherheit klassifizierbar. Interessanterweise ist die Genauigkeit der Klasse > 1530 mit 0.83 höher als für die Klasse ∞ mit 0.80, obwohl die Sicherheit der korrekten Klassifizierungen mit 0.21 im

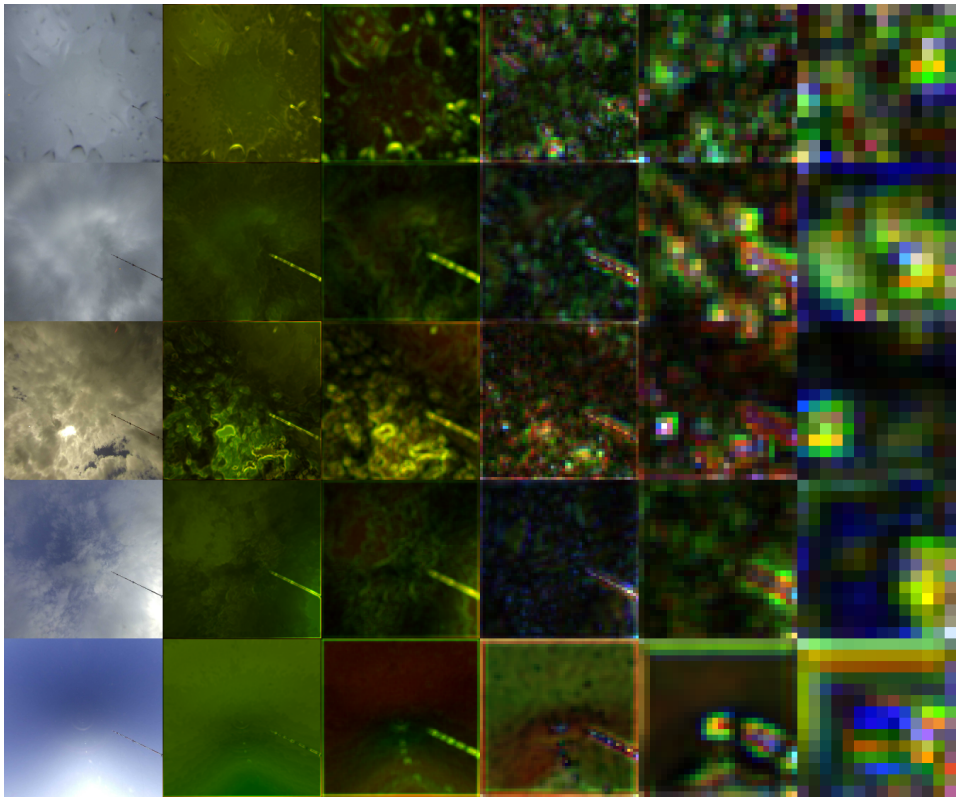

Figure 3.4: VGG16 Aktivierungen. Pro Spalte sind Aktivierungen für *conv. layer* gleicher Größe in Farbkanälen *RGB* oder *RG* dargestellt. Dabei zeigt jeder Farbkanal die *min-max* normalisierte mittlere Aktivierung der *feature maps* von einem *conv. layer*. Für jede Höhenklasse 0 bis 4 wird eine Reihe dargestellt.

Vergleich zu 0.13 geringer ist. Dies deutet darauf hin, dass Bilder der Klasse > 1530 mehr Merkmale anderer Klassen tragen als die Extrema und dennoch mit hoher Genauigkeit unterschieden werden können.

3.3.1 Aktivierungen des Netzes

Die *convolutional layer* zeigen abhängig von dem Eingabebild unterschiedliche Muster in Aktivierungen. Die Aktivierungen in Abbildung 3.4 sind die einzelnen Farbkanäle für eine klarere Visualisierung *min-max* normalisiert. Die Spalten 1 und 2 zeigen die Aktivierungen der unveränderten *convolutional layer* in *RG* Farbkanälen. Vor allem Kanten und Umrisse im Bild erhalten hohe Aktivierungen. Alle Elemente des Bildes, sowie der Wettermast, werden gleichmäßig erkannt und es werden bereits Muster sichtbar. Hingegen sind in den trainierten *layers* viele lokal konzentrierte Aktivierungen zu erkennen, die auf eine Wolkenerkennung hindeuten. So treten im Bild mit klarem Himmel wenige Konzentrationspunkte auf. In der letzten Spalte sind Wolkenmuster nicht mehr erkenntlich, aber Konzentrationspunkte aus vorherigen *layers* scheinen sich zu übertragen.

3.3.2 Mögliche Verbesserungen

Die *accuracy* des Modells von 0.76 ist deutlich höher als zufällige Vorhersagen mit 0.20. Jedoch stellt die Höhenklassifikation mit klaren Unterschieden in den Klassen keine besondere Herausforderung dar, sodass eine höhere Genauigkeit zu erwarten wäre. Einerseits könnten mehr Daten die Leistung des Modells verbessern. Hierzu können weitere *augmentation* Methoden, wie bspw. zufällige Größenänderung durch *cropping* oder Rotationen der Bilder, verwendet werden. Die Diversität der Wolkenbilder ist gering, da von einzelnen Tagen stammen. Würden mehr Daten zur Verfügung stehen, könnten verschiedenartige Wolkenbilder selektiert werden, die mit mehr Diversität bessere Generalisierungen erlauben. Zudem könnte eine zufällige Datenreihenfolge für jede Epoche, durch aufteilten des *train* Datensatzes in mehrere Dateien, hilfreich beim Training sein.

Da der Regen und die Sonneneinstrahlung mit der Wolkenhöhe zusammenhängen, bieten sich die hierfür verfügbaren Daten für Vorhersagen, in Form von unterschiedlichen *tasks* des Modells, an. Dies könnte verbesserte Wolkenhöhenvorhersagen bewirken.

Bibliography

[Alex Krizhevsky and Hinton(2012)] Ilya Alex Krizhevsky, Sutskever and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012. URL `http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf`.

[Frossard(2016)] Davi Frossard. Vgg in tensorflow, 2016. URL `https://www.cs.toronto.edu/~frossard/post/vgg16/`.

[Karen Simonyan(2014)] Andrew Zisserman Karen Simonyan. Very deep convolutional networks for large-scale image recognition. *CoRR*, 2014. URL `http://arxiv.org/abs/1409.1556`.

[Lange(2015)] Ingo Lange. Kürzelliste wettermast, 2015. URL `https://wettermast.uni-hamburg.de/Downloads/Kurzelliste_Wettermast.pdf`.

[Lange(2016)] Ingo Lange. Verwendung der kamera vivotek fe8174v als wolkenkamera, 2016. URL `https://wettermast.uni-hamburg.de/Downloads/Wolkenkamera_FE8174V.pdf`.

[Lange(2018)] Ingo Lange. Daten vom wettermast hamburg, 2018. URL `https://wettermast.uni-hamburg.de/Downloads/Wettermast_Datenbeschreibung.pdf`.

Anhang

Verwendete Software/Hardware

- **python3** mit numpy
- **pysolar** für die Sonnenwinkelberechnung
- **matplotlib** für Visualisierungen
- **opencv-python** für Bildverarbeitung
- **TensorFlow**, **TensorBoard**
- **cluster** des Bereichs: Wissenschaftliches Rechnen (slurm jobs)
- **git** für Verwaltung des Projektcodes
- Bericht erstellt mit LaTeX