

Projektarbeit ADS SunSpotter

Applied Data Science (ADS) FS 2021

Matthias Christen, Claudio Hauser, Marion Mürner

Gruppe 01

Agenda

- Einleitung (Hintergrund, Problemstellung, Zielsetzung, Forschungsfrage)
- Methode und Vorgehensweise
- Diskussion & Ergebnisse
 - Hilfstools
 - Datenquellen
 - Scraping-Tools
 - Model bauen und trainieren
 - Model Performance
 - Demo SunSpotter Applikation
 - Systemarchitektur – Integration in Clientsoftware
- Ethische Fragestellungen
- Schlussfolgerungen

Einleitung (Hintergrund, Problemstellung, Zielsetzung, Forschungsfrage)

Hintergrund

- Die Alpen sind ein beliebter Freizeitort und ziehen jährlich mehrere tausend Bergsportbegeisterte an. Das Vergnügen und die Sicherheit hängen stark vom Wetter ab.

Problemstellung

- Lokale Wetterprognosen im Alpenraum sind aber schwierig zu berechnen und anstatt mit Sonne wird man mit Nebel oder Regen überrascht

Zielsetzung

- Wo ist es **jetzt** schön → Zusatzinformation zur Prognose für Freizeitaktivitäten (Bergsport, Gleitschirmfliegen, Segelflugsport)
- Gesamtübersicht fehlt → manuelles durchklicken von Webcams nötig
- Maschinelles Auswerten von Webcam-Bilddaten im Alpenraum, um die lokale aktuelle Wetterlage auszuwerten und davon eine Gesamtübersicht zu erstellen

Forschungsfrage

- Kann die Data Science Technologie Bildklassifikation verwendet werden, um Webcam-Bilddaten auszuwerten und damit eine zuverlässige Aussage über die Wetterlage am Webcam Standort zu erstellen?



Figure 1: Webcam: Bayerische Zugspitzbahn

Methode und Vorgehensweise (1)

Daten untersuchen und verstehen

- Internetrecherche nach Webcams

Daten sammeln bereinigen und labeln

- Scraping (Selenium, WebcamGet, eigener Scraper in JS)
- Labeling via Ordnerstruktur (3 Klassen sunny, cloudy, rainy)

Modell bauen, trainieren und testen (compile, fit and predict)

- Jupyter-Notebook
- Eigenes CNN und Klassifizierung
- Pre-trained Model für CNN, eigne Klassifizierung
- AWS Sagemaker

Modell verbessern und Prozess wiederholen

- Verschiedene Ansätze, Optimierung und Pre-trained Modelle ausprobieren

Model Deployment

- Verschiedene Ansätze versucht: Tensorflow.js, Java-Spark-API, Sagemaker **all fail** Python-Script **win!**

Integration in Clientanwendung

- Scraping von aktuellen Webcambildern
- Geo-Tagging auf Karte
- Prediction ausführen und Ergebnisse auf Karte anzeigen

End-to-end machine learning pipeline

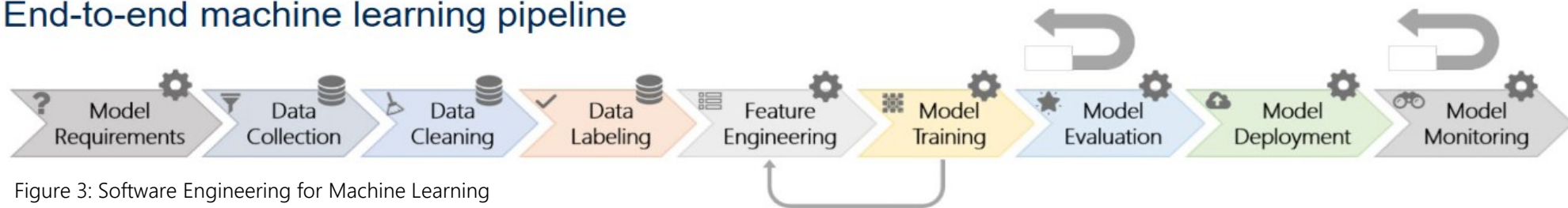
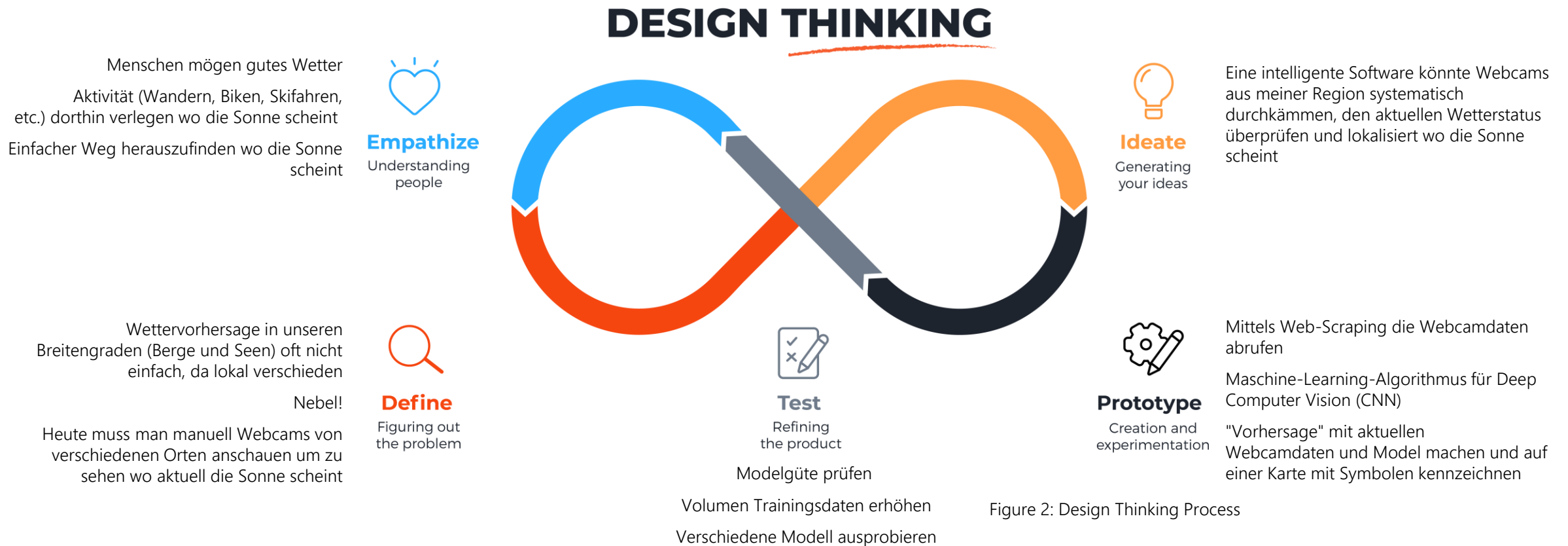


Figure 3: Software Engineering for Machine Learning

Methode und Vorgehensweise (2)



Diskussion & Ergebnisse – Hilfstools

Tensorboard:

- Visualisierungs-Toolkit von TensorFlow.
- Visualisierung des Modellgraphen und Metriken für Loss und Accuracy.



Amazon Sagemaker:

- Cloubasierte Plattform für ML-Algorithmen.
- Intanziierung des Klassifikationsmodells auf Amazon-Cloud (AWS – Sagemaker).
- Ermöglicht ressourcenaufwändiges Training vom Klassifikationsmodell.



Docker:

- Docker Container für Sunspotter Webapp und TensorBoard



Git/Github:

- Repository für gemeinsame Entwicklung:
<https://github.zhaw.ch/muon/ads-fs2021-project-sunspotter>



Diskussion & Ergebnisse – Datenquellen

Windy.com:

- Webcam-Bilddaten
- API zur Benutzung der Webcam-Bilddaten auf eigener Homepage.
- Uneingeschränkte API-Funktion kostenpflichtig.

Foto-webcam.eu:

- Webcam-Bilddaten
- Timestamp in der URL

Openrouteservice:

- Daten für Geotagging
- Max. 1000 Anfragen / Tag, max. 100 Anfragen / min.

Diskussion & Ergebnisse – Scraping (1)

Selenium:

- Framework für automatisierte Softwaretests von Webanwendungen.
- Anwendung: Interaktion mit HTML Elementen, Web-Formulare, Drag & Drop, Navigation zwischen Webseiten etc.
- Limitation: Bild-Dateien im Filesystem speichern.



WebcamGet 2.0.3:

- Deployment: Heise Download, Sascha Löffler
- Speichert Webcam-Bilder auf der Festplatte. Webcam-URL, Speicherpfad & Scraping-Intervalle für mehrere Webcams konfigurierbar.
- Limitation: Scraping von aktuellen Aufnahmen / Kein Zugriff auf Webcam-Bilder aus der Vergangenheit.
- Windy.com

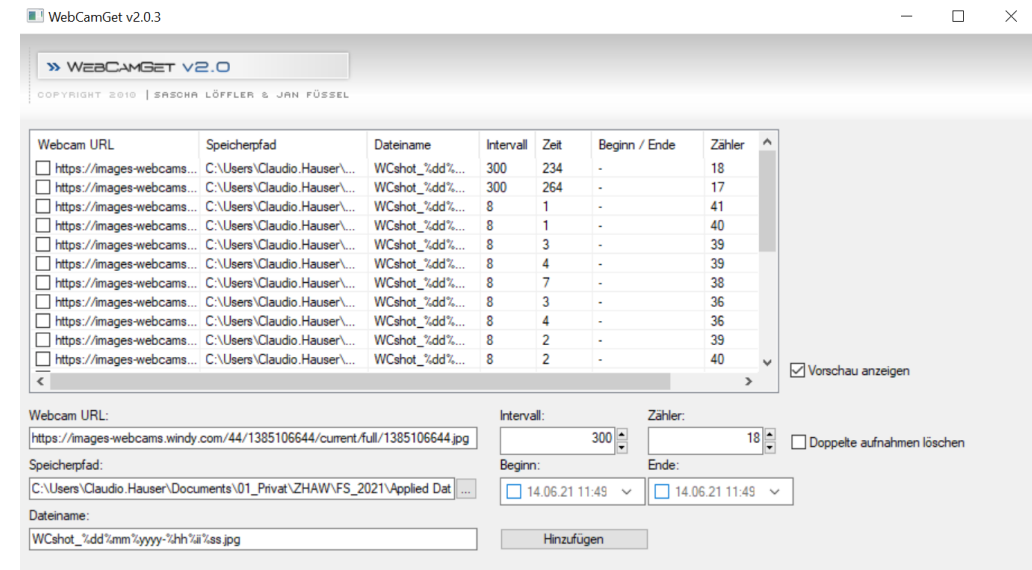


Figure 4: WebcamGet User Interface

Diskussion & Ergebnisse – Scraping (2)

Eigener Scraper in JS:

- Eigenentwicklung für das Scraping von Webcam-Bilder von Foto-webcam.eu.
- Scraping von Bildern aus der Vergangenheit möglich (Timestamp in URL: https://www.foto-webcam.eu/webcam/%webcamid/%YYYY/%mm/%dd/%HH%MM_la.jpg).
- User Interface für die Sucheingabe: URL, Datum, Intervalle, Speicherpfad etc.
- Foto-webcam.eu

Download webcam images

Datetime from: Datetime to: Time interval in hours:

Webcam ID:

Webcam url:

[Webcam map](#)

Figure 5: Eigener Scraper User Interface

Diskussion & Ergebnisse – Model bauen, trainieren und optimieren

Bildklassifizierung mit Convolutional Neural Network (CNN)

- Verwendung von Tensorflow und Keras in Jupyter-Notebook
- Ansatz 1: Trainieren eines eigenen CNN
--> *ownCNNClassificationSunSpotter.ipynb*
- Ansatz 2: Einsetzen von Transfer Learning. Verwenden eines vorgeladenen head-less Models von TensorflowHub und nur Klassifizierung selber trainieren
--> *transferLearningCNNClassificationSunSpotter.ipynb*

Optimierungen und Regulierungen

Wenig Trainingsdaten

- Data Augmentation

Vanishing/Exploding Gradients

- Optimize Gradient Descent (Gradually reduce learning rate with optimizer Adam)
- Kernel_initilaizier (he_normal)
- Activation Function (relu)

Vermeiden von Overfitting

- Training abbrechen, wenn die Validierung am Minimum ist durch Early Stopping
- Dropout Layer (dropout rate auf 0.2)
- Cost-Function L2 Regularizer, damit irrelevante Verbindungen deaktiviert werden

Model: "sequential_7"			Model: "sequential"		
Layer (type)	Output Shape	Param #	Layer (type)	Output Shape	Param #
sequential_3 (Sequential)	(None, 180, 180, 3)	0	keras_layer (KerasLayer)	(None, 1792)	4363712
resizing_6 (Resizing)	(None, 180, 180, 3)	0	dense (Dense)	(None, 128)	229504
rescaling_9 (Rescaling)	(None, 180, 180, 3)	0	dropout (Dropout)	(None, 128)	0
conv2d_19 (Conv2D)	(None, 180, 180, 16)	448	dense_1 (Dense)	(None, 64)	8256
max_pooling2d_19 (MaxPooling)	(None, 90, 90, 16)	0	dropout_1 (Dropout)	(None, 64)	0
conv2d_20 (Conv2D)	(None, 90, 90, 32)	4640	dense_2 (Dense)	(None, 3)	195
max_pooling2d_20 (MaxPooling)	(None, 45, 45, 32)	0	Total params: 4,601,667		
conv2d_21 (Conv2D)	(None, 45, 45, 64)	18496	Trainable params: 237,955		
max_pooling2d_21 (MaxPooling)	(None, 22, 22, 64)	0	Non-trainable params: 4,363,712		
conv2d_22 (Conv2D)	(None, 22, 22, 128)	73856			
max_pooling2d_22 (MaxPooling)	(None, 11, 11, 128)	0			
dropout_2 (Dropout)	(None, 11, 11, 128)	0			
flatten_6 (Flatten)	(None, 15488)	0			
dense_13 (Dense)	(None, 128)	1982592			
dense_14 (Dense)	(None, 64)	8256			
dense_15 (Dense)	(None, 3)	195			
Total params: 2,088,483					
Trainable params: 2,088,483					
Non-trainable params: 0					

Figure 6: Model summary (eigenes CNN-Model
4 Convolution Layers)

Figure 7: Model summary (transfer learning)

Diskussion & Ergebnisse – Model Performance

Use head-less pretrained Model for CNN

We tried the following models:

- Use trainable=False to freeze the variables in the feature extractor layer, so that the training only modifies the new classifier layer.
- Use trainable=True to fine tune the variables in the feature extractor layer.

Datenvolumen

1312 Bilder a 3 Klassen für Training
327 Bilder a 3 Klassen für Validierung

"Model"	"Fine tuned?"	"Training loss"	"Training accuracy"	"Validation loss"	"Validation accuracy"	"Comments"
resnet_v2_50	yes	0.7248	0.9413	0.8801	0.8594	generalizes bad (always 100% confidence in one class)
resnet_v2_50	no	0.8994	0.5831	0.8569	0.5375	generalizes bad (always 100% confidence in one class)
inception_resnet_v2	yes	1.0882	0.8742	1.8426	0.5250	generalizes bad (always 100% confidence in one class)
inception_resnet_v2	no	0.6287	0.8308	0.9929	0.6219	generalizes bad (always 100% confidence in one class)
mobilenet_v2_140_224	yes	0.5639	0.9345	1.0495	0.6125	second best results
mobilenet_v2_140_224	no	0.6338	0.9345	0.7113	0.7750	best validation loss & generalizes best

Own CNN-Model with 5 convolution layers

"Model"	"Fine tuned?"	"Training loss"	"Training accuracy"	"Validation loss"	"Validation accuracy"	"Comments"
selfmade model	-	0.3565	0.8442	0.3142	0.8592	lowest loss and high accuracy but generalizes not good


Figure 8: Vergleich loss & accuracy


Würdigung Klassifizierung

- Modell fitted gut mit den Trainingsdaten aber generalisiert mittelmässig
- Modell-Generalisierung durch mehr Trainingsdaten stärken


Demo SunSpotter

SunSpotter







4's Hotel Lindenwirt
☀️ Sunny (47.20%), 15.06.2021 07:20:40
[Scrape](#)




Adlersruhe / Blick nach Westen zum Großglockner
☁️ Cloudy (35.94%), 15.06.2021 07:19:53
[Scrape](#)




Alpenhotel Ratsberg
☀️ Sunny (44.13%), 15.06.2021 07:19:20
[Scrape](#)




Alta Badia / San Cassiano
☀️ Sunny (44.89%), 15.06.2021 07:19:48
[Scrape](#)




Ankogel Bergbahnen / Mittelstation
☀️ Sunny (43.58%), 15.06.2021 07:19:56
[Scrape](#)




Ankogel Bergbahnen / Mittelstation
☀️ Sunny (44.23%), 15.06.2021 07:20:28
[Scrape](#)



Apparthotel Germania
☀️ Sunny (41.65%), 15.06.2021 07:20:32
[Scrape](#)



Astental / Burgstalleralm
☀️ Sunny (43.47%), 15.06.2021 07:20:11
[Scrape](#)



Astental / Sadnighaus
☀️ Sunny (44.37%), 15.06.2021 07:20:33
[Scrape](#)

1. Scrape webcams 2. Geocode all webcams ☐ Use geocode service 3. Predict all webcams [Open Tensorflow board](#)

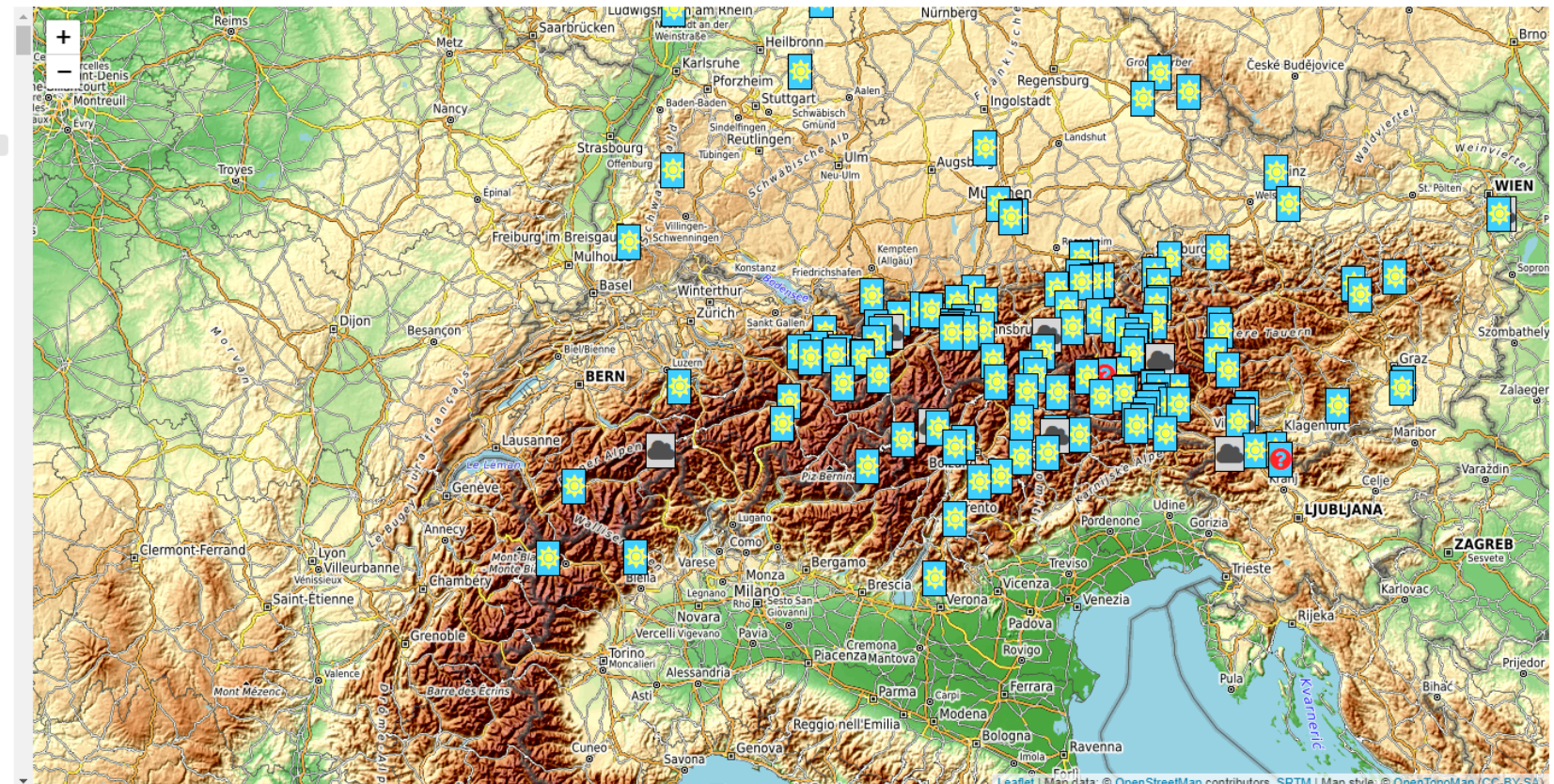


Figure 9: SunSpotter Webapp

Diskussion & Ergebnisse – Integration in Clientsoftware

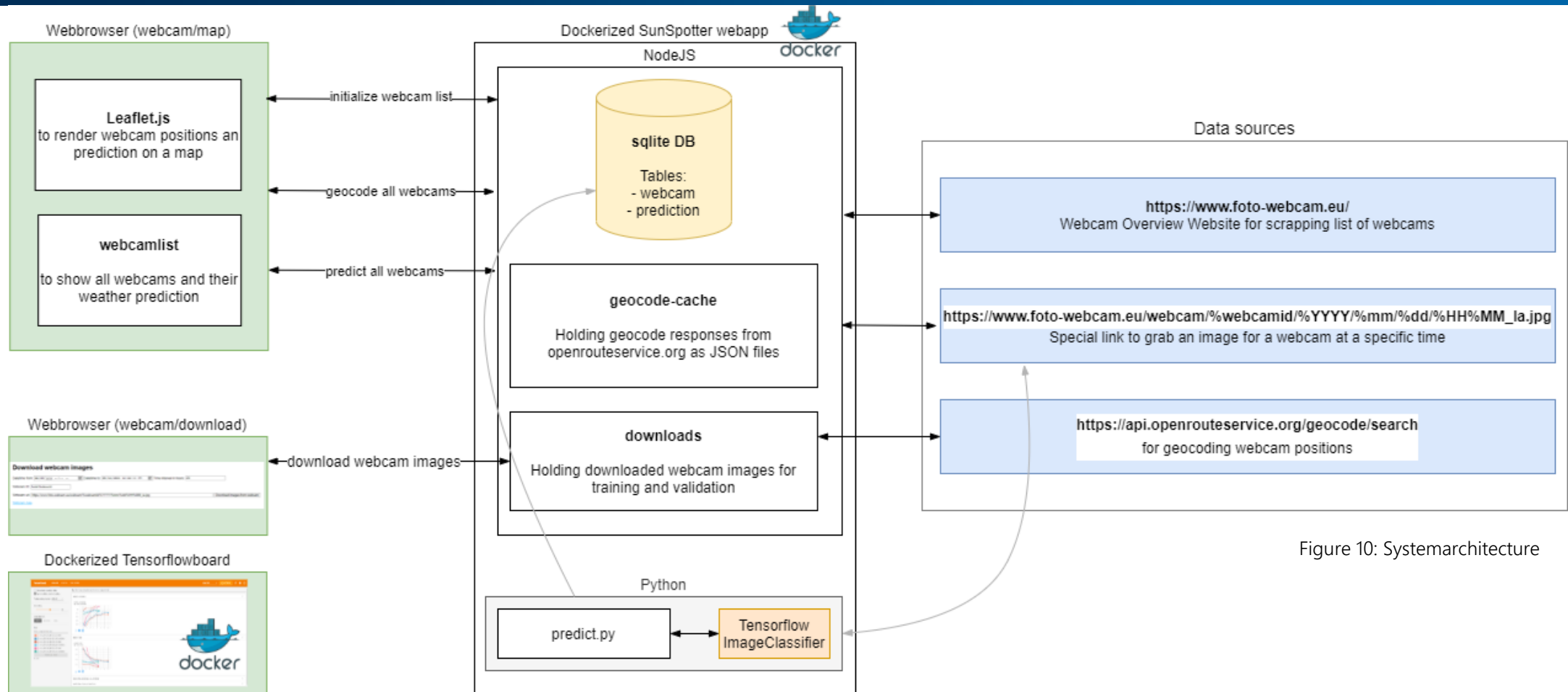


Figure 10: Systemarchitecture

Ethische Fragenstellung

Persönlichkeitsschutz

- Bildnisrecht – Jeder Mensch darf selbst bestimmen, ob und wie Bilder von ihm verwendet und veröffentlicht werden.
- Problemstellung: SunSpotter verwendet u. U. Bilder auf denen Personen zu sehen sind.
- Mittels Object Detection Personen erkennen und entweder Bild nicht verwenden, oder Person retuschieren.



Urheberrecht

- Bildrechte – Die Urheberrechte der Webcambilder müssen beachtet werden.
- Bsp. Foto-Webcam: „Die Inhalte und Bilder unserer Website unterliegen - sofern nicht anders gekennzeichnet - unserem Urheberrecht und dürfen ohne vorherige schriftliche Zustimmung weder als Ganzes noch in Teilen verbreitet, verändert oder kopiert werden.“ (Datenschutz - Foto-Webcam, <https://www.foto-webcam.eu/impressum/m.eu>)
- Bei öffentlichem Deployment, oder kommerzieller Nutzung muss die Berechtigung eingeholt werden.



Sonstige ethische Fragestellungen

- SunSpotter ist mit Ausnahme der persönlichkeits- und urheberrechtlichen Punkten ethisch unbedenklich.

Schlussfolgerungen

- Trainingsdatenmenge ist theoretisch beliebig gross für Training (z.B. stündliches Modell-Training mit Live Webcam-Bilddaten).
- Trainingsdaten benötigen viel Speicherkapazität und das Model-Training dauert länger.
- Labeln der Bilder ist sehr aufwändig.
- Ansatz mit Transfer Learning ist sehr hilfreich.
- "Pröbeln" gehört dazu.
- Rechenpower absolut notwendig, da ansonsten viel zu viel Zeit für die Trainings gebraucht wird.
- Integration in die Clientanwendung ist nicht trivial.
- Forschungsergebnis: Webcam-Bilder lassen sich mit ML-Algorithmen klassifizieren und ermöglichen Aussagen zur aktuellen Wetterlage. Die Generalisierbarkeit der Klassifizierung hat noch Verbesserungspotenzial.

Literatur & Datenquellen

Literatur

- Géron A. (2017). Hands-On Machine Learning with Scikit-Learn & TensorFlow. First Edition. United States of America: O'Reilly.
- TensorFlow community (2021). Convolutional Neural Network (CNN). Abgerufen von <https://www.tensorflow.org/tutorials/images/cnn>
- NodeJS (2021), <https://nodejs.org/en/> (Abgerufen am 14.06.2021)
- SQLite Datenbank (2021), <https://sqlite.org/index.html> (Abgerufen am 14.06.2021)

Datenquellen

- Foto-webcam.eu (2021), Webcam-Bildaten, <https://www.foto-webcam.eu/> (Abgerufen am 14.06.2021)
- Windy.com (2021), Webcam-Bilddaten, <https://www.windy.com/-Webcams/webcams/> (Abgerufen am 14.06.2021)
- Openrouteservice (2021), Geocoding service, <https://openrouteservice.org/> (Abgerufen am 14.06.2021)
- Tensorflow (2021), mobilenet_v2_140_224 Image classification model, https://tfhub.dev/google/imagenet/mobilenet_v2_140_224/feature_vector/4 (Abgerufen am 15.06.2021)

Abbildungsverzeichnis

- Figure 1: Webcam: Bayerische Zugspitzbahn (2021): <https://www.foto-webcam.eu/webcam/zugspitze-ost/> (Abgerufen am 14.06.2021)
- Figure 2: Design Thinking Process: MAQE <https://www.maqe.com/insight/the-design-thinking-process-how-does-it-work/> (Abgerufen am 13.06.2021)
- Figure 3: Software Engineering for Machine Learning: A Case Study (2019 Amersi et al. (Microsoft))
- Figure 4: WebcamGet User Interface: HEISE <https://www.heise.de/download/product/webcamget-83284> (Abgerufen am 14.06.2021)
- Figure 5: Eigener Scraper JS User Interface, Printscreen out of Browser
- Figure 6: Model summary (own CNN-Model), Printscreen out of Jupyter-Notebook
- Figure 7: Model summary (transfer learning), Printscreen out of Jupyter-Notebook
- Figure 8: Vergleich loss & accuracy, Printscreen out of Jupyter-Notebook
- Figure 9: SunSpotter Webapp, Printscreen Browser
- Figure 10: Systemarchitecture, Own drawing made with draw.io