

PR Software Praktikum - WS 2021

Rock.AI - Backend & ML Dokumentation

21 Jänner 2022

Stefan Hangler, 11904159
Marie Moser-Schwaiger, 11915381

Inhaltsverzeichnis

| | |
|--|-----------|
| Verwendete Entwicklungsplattform: Firebase | 2 |
| Verwendete Datenbank: Firestore & Storage | 2 |
| Datensätze von API, Firebase speicher & ML Labels erstellen | 2 |
| API_Request_generate_JSON.py | 2 |
| uploadJSONFileToFirestore.py | 2 |
| Image Augmentation (Bilddatenerweiterung) | 3 |
| image_augmentation.py | 3 |
| Verwendeter Service: AutoML Vision Edge | 4 |
| Einrichten des Machine Learnings | 4 |
| Einrichten eines exportierbaren Modells | 4 |
| Einrichten eines Modells für Online-Nutzung | 15 |
| Trainierte Modelle und Ergebnisse | 17 |
| Modell: rockAI_ml | 17 |
| Modell: rockAi_ml_multi_label | 18 |
| Modell: rockAi_singleLabels2 | 19 |
| Modell A: Edge-basiert | 19 |
| Modell B: Cloud-basiert | 20 |
| Modell: rockAI_singleLabel_augmented | 21 |
| Fazit | 24 |
| Mögliche zukünftige Arbeiten | 24 |

Verwendete Entwicklungsplattform: [Firebase](#)

Firebase ist eine umfassende Entwicklungsplattform zur Entwicklung von mobilen und webbasierten Apps in der Google Cloud Plattform. Auf der Plattform wird eine breite Auswahl an Services zur Verfügung gestellt.

Verwendete Datenbank: Firestore & Storage

Cloud Firestore ist eine NoSQL-Dokumentendatenbank, mit der Daten für Mobil- und Webanwendungen einfach gespeichert, synchronisiert und abgefragt werden können.

[Cloud Firestore | Store and sync app data at global scale | Firebase](#)

Aufgesetzt wurde die Firebase Firestore Datenbank und der Firebase Storage für die Bilder mit der allgemeinen Dokumentation von Firebase: <https://firebase.google.com/docs/firestore>

→ wichtig für Nutzung von AutoML: Bucket muss gleiche Konfigurationen wie AutoML Bucket haben. Sprich: *Location: us-central1, required location type: Region, required storage class: Standard*

→ Bilder in den Bucket laden, in Ordner images/

Datensätze von API, Firebase speicher & ML Labels erstellen

Es werden zwei selbstgeschriebene Python-Programme verwendet, um die Datensätze von der API <https://health.api.makia.ml> zu holen und in einem benutzerdefinierten Format in die Firebase Firestore zu speichern (Bilder werden in den Firebase Storage gespeichert).

[API_Request_generate_JSON.py](#)

Dies Datei holt sich zuerst die Datensätze von der oben genannten URL als JSON-File und speichert die Daten in lokale Listen und Variablen ab. Anschließend werden die zugehörigen Bilder in einem lokalen Ordner gespeichert und die in die Firebase Storage hochgeladen. Um in der Firestore Datenbank später die Bilder zu hinterlegen, wird von jedem Bild der https-Token ebenso lokal abgespeichert. Abschließend generiert das Skript ein JSON-File nach einem benutzerdefinierten Format, für die gewünschte Struktur in der Firestore Datenbank.

Ebenso wird eine CSV-Datei erstellt, welche alle Bilder für das Training beim Machine Learning labelt. (siehe [Schritt 1 unter Einrichten eines exportierbaren Modells](#))
Genauere Dokumentationen befinden sich in dem [Python-File](#).

[uploadJSONFileToFirestore.py](#)

Mit dem Befehl:

```
python uploadJSONFileToFirestore.py <json-file-name>.json add <database-name>
```

wird jedes JSON-File (anzugeben in <json-file-name>), welches dem Format für ein Firestore Upload entspricht, in die Firestore Datenbank unter einem gewünschten Datenbanknamen hochgeladen (anzugeben bei <database-name>).

Wichtig dabei ist das die Files *service-account.json* und *google-services.json* von der zugehörigen Firebase Datenbank stammen und die Zugriffsrechte richtig zugewiesen sind.

Diese zwei Files ermöglichen ein automatisches Hochladen und anpassen von den API-Datensätzen und der Firestore Datenbank. Somit können zukünftig größere Mengen an Datensätze ohne Zeitaufwand von der API in die Firestore Datenbank hochgeladen werden.

Ebenso kann die Struktur der Datenbank leicht durch das anzupassende JSON-File in [API_Request_generate_JSON.py](#) adaptiert werden, um aufkommende Probleme schnell zu beheben (sei es für das ML oder auch für die App aufkommende Probleme).

Image Augmentation (Bilddatenerweiterung)

Die Bilddatenerweiterung ist eine Technik, mit der die Größe eines Trainingsdatensatzes künstlich vergrößert werden kann, indem modifizierte Versionen von Bildern im Datensatz erstellt werden.

Das Training von Deep Learning-Modellen für neuronale Netze auf mehr Daten kann zu besseren Modellen führen, und die Augmentierungstechniken können Variationen der Bilder erstellen, die die Fähigkeit der angepassten Modelle verbessern, das Gelernte auf neue Bilder zu verallgemeinern.

image_augmentation.py

In diesem File werden alle lokal gespeicherten originalen Bilder bearbeitet. Dabei werden von der Library [imgaug](#) (siehe auch [Dokumentation](#)) ein paar standard Veränderungen an den Bildern vorgenommen um sie zu modifizieren, wie zum Beispiel: horizontales spiegeln, rotieren, Kontrast Veränderungen oder Blurring.

Diese Bilder werden anschließend lokal gespeichert und können anschließend manuell in die Firebase Firestore Datenbank hochgeladen werden unter `/images_augmentation` um den bestehenden Datensatz zu erweitern.

Um diese "neuen" Bilder anschließend für das Machine Learning zu labeln, setzt man in dem File [API_Request_generate_JSON.py](#) die Variable `GENERATE_LABELS_AUGMENTATION_IMAGES = True` und somit werden zu den original Bildern auch die augmentierten Bilder gelabelt.

Weitere Referenzen findet man in dem File [References Image Augmentation for Machine Learning](#), dabei werden ein paar interessant Paper und Code Snippets angeführt, welche für weitere Arbeiten interessant sein könnten. Ebenso gibt es auch die Möglichkeit, Librarys von Tensorflow oder PyTorch zu verwenden, welche ebenfalls in diesem PDF angeführt werden. Ebenso findet man einen Link zu einem Deep Learning Augmenter, welcher anscheinend für sein Machine Learning Model die besten Augmentierung bestimmen kann, was ebenfalls für zukünftige Arbeiten interessant und hilfreich sein könnte.

Verwendeter Service: AutoML Vision Edge

Für das Machine Learning wurde die Machine Learning Funktion bei Firebase genutzt. In diesem Fall wurde mit dem Image-Labeling mit AutoML, ein einfaches Labeling der Bilder mit dem Attribut "Gefährdungsklasse" durchgeführt.

Dokumentationen von Google: [Firebase Machine Learning](#)

AutoML Vision Edge ist ein Google-Cloud Service, die Nutzung ist nur mit Bezahlung möglich. Es gibt dafür eine Testversion, diese wurde in diesem Fall verwendet. Bei der Testversion wird ein Budget von 300\$ (~260€) für 90 Tage bereitgestellt. Damit folgt dann ein Upgrade des Firestore-Projekt auf den Blaze-Tarif (pay as you go).

Informationen zu AutoML Vision Edge: [AutoML](#)

Informationen zu den Tarifen von Firebase: [Firebase Pricing](#)

Einrichten des Machine Learnings

Einrichten eines exportierbaren Modells

Für das Einrichten des ML wurden die Schritte der folgenden Dokumentation gefolgt.

Dokumentation von Google: [Train an image labeling model with AutoML Vision Edge](#)

1) Trainingsdaten zusammenstellen

- a) Rock.AI-Bilddaten über API in Cloud Storage hochgeladen
- b) .csv Datei mit Bild-Link und passendem Label muss in den **gleichen Bucket wo sich die Bilder befinden** geladen werden

```

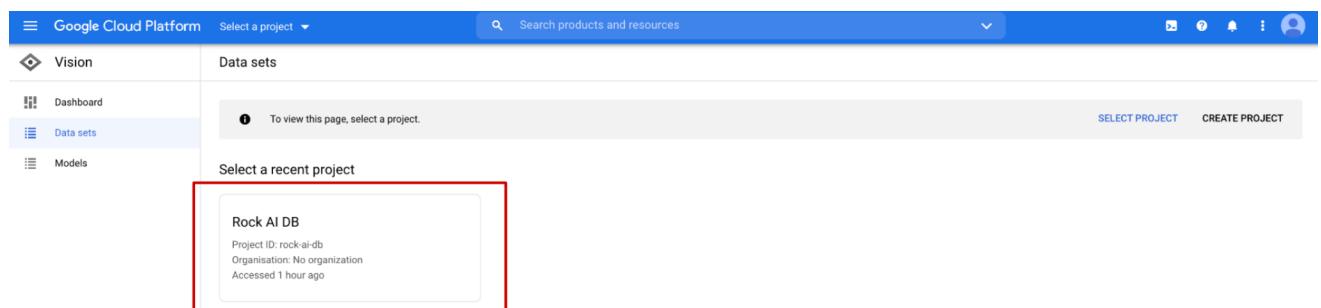
1  gs://rock-ai-us-central1/images/image00_03.jpeg,HOCH
2  gs://rock-ai-us-central1/images/image00_04.jpeg,HOCH
3  gs://rock-ai-us-central1/images/image00_05.jpeg,HOCH
4  gs://rock-ai-us-central1/images/image00_06.jpeg,HOCH

```

- c) weitere Infos über Datenvorbereitung: [Preparing your training data](#)

2) Trainieren

- a) In "Vision Data sets" in der Google Cloud Console das gewünschte Projekt auswählen.



- b) Auf **New dataset** klicken

The screenshot shows the Google Cloud Platform Vision Data sets page. It lists three existing datasets: 'rockAI_singleLabel2' (Single-Label Classification), 'rockAI_ml_multiLabel' (Multi-Label Classification), and 'rockAI_ml' (Single-Label Classification). A 'NEW DATA SET' button is located at the top right of the list.

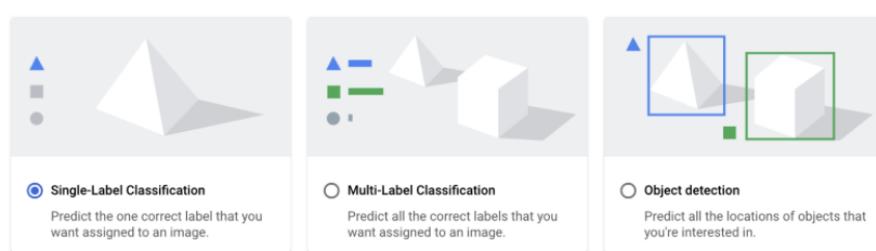
Dem neuen Datensatz einen Namen geben, Single-Label Classification auswählen und auf **Create dataset** klicken.

Create new data set

Data set name *

Use letters, numbers and underscores up to 32 characters.

Select your model objective



CANCEL

CREATE DATASET

- c) Im **IMPORT** Tab, die Auswahl **Select a CSV file on Cloud Storage** wählen und die hochgeladene .csv Datei im Cloud Storage auswählen

The screenshot shows the 'IMPORT' tab of the 'test123' dataset. In the 'Select files to import' section, 'Select a CSV file on Cloud Storage' is selected. Below it, the 'Select a CSV file on Cloud Storage' section shows a 'gs://...' input field and a 'BROWSE' button. To the right, a sidebar titled 'Select object' shows a file tree with 'single_labels220111.csv' highlighted. A large red arrow points from this file to the 'BROWSE' button.

The screenshot shows the 'IMPORT' tab of the 'test123' dataset. It displays the message 'Running: Importing images'. In the bottom right corner, there are 'SELECT' and 'CANCEL' buttons, with a red arrow pointing to the 'SELECT' button.

Nachdem der Import der Bilder vervollständigt ist, werden diese im **IMAGES** Tab mit den zugehörigen Labels angezeigt. Es kann ein Problem beim Hochladen und Labeln mancher Bilder auftreten, diese werden dann als "unlabelled" angezeigt und können direkt selbst gelabelt werden.

The screenshot shows the Google Cloud Vision interface for a dataset named 'test123'. On the left, the sidebar lists 'Dashboard', 'Data sets', and 'Models'. The main area has tabs for 'IMPORT', 'IMAGES', 'TRAIN', 'EVALUATE', and 'TEST & USE'. The 'IMAGES' tab is selected. A table shows the count of images: 'All images' (439), 'Labelled' (436), and 'Unlabelled' (3). Below this is a 'Filter' section with 'Unlabelled' selected. Three small images of rocks are shown. A red arrow points from the 'Unlabelled' table entry to a larger image detail view on the right. This detail view shows a close-up of a rock surface with moss and is titled 'Image 1 of 3'. It includes a 'Filter labels' dropdown where 'SEHR_GERING' is selected. The image itself is labeled 'gs://rockair-us-central1/images/image62_3.jpg'.

Nach dem labeln aller Bilder sieht der **IMAGES** Tab folgendermaßen aus:

This screenshot shows the same 'test123' dataset after labeling. The 'IMAGES' tab now displays 439 labeled images. The 'Unlabelled' row in the table shows 0 images. Below the table, a grid of 15 images is shown, each with its label: GERING(1), GERING(1), GERING(1), MITTEL(1), HOCH(1), GERING(1), MITTEL(1), HOCH(1), GERING(1), GERING(1), MITTEL(1), GERING(1), GERING(1), GERING(1), MITTEL(1), MITTEL(1). At the bottom, there are navigation controls for 'Images per page' (50) and '1 – 50 of many'.

d) Im TRAIN Tab auf **Start training** klicken

The screenshot shows the Google Cloud Platform Vision interface for a dataset named 'test123'. The 'TRAIN' tab is selected. Below it, there's a table showing image counts for four categories: GERING, HOCH, MITTEL, and SEHR_GERING. At the bottom, a large blue 'START TRAINING' button is highlighted with a red box.

| Labels | Images | Train | Validation | Test |
|-------------|--------|-------|------------|------|
| GERING | 197 | 157 | 20 | 20 |
| HOCH | 73 | 59 | 7 | 7 |
| MITTEL | 130 | 104 | 13 | 13 |
| SEHR_GERING | 39 | 31 | 4 | 4 |

a) Dem Model einen Namen geben und **Edge** als Möglichkeit wählen.

The screenshot shows the 'Train new model' wizard. Step 1: Define your model. It asks for a model name ('test123_191122') and offers two options: 'Cloud hosted' and 'Edge'. The 'Edge' option is selected and highlighted with a red box. A 'CONTINUE' button is at the bottom.

b) Die gewünschte Optimierung wählen, in diesem Fall wurde **Best trade-off** gewählt

The screenshot shows the 'Train new model' wizard. Step 2: Optimise model for. It lists three goals: 'Higher accuracy', 'Best trade-off', and 'Faster predictions'. The 'Best trade-off' option is selected and highlighted with a red box. A note below says: 'Please note that prediction latency estimates are for guidance only. Actual latency will depend on your network connectivity.' A 'CONTINUE' button is at the bottom.

c) Das Budget für die Knotenstunden wählen

Train new model

Define your model
 Optimise model for
 ③ Set a node hour budget

Enter the maximum number of node hours you want to spend training your model.

We recommend using [3 node hours](#) for your dataset. However, you can train for as little as 1 node hours. You may also eligible to train with free node hours. [Pricing guide](#)

Set your budget *
 node hours
Estimated completion date: Jan 19, 2022 2 pm
GMT+1

START TRAINING CANCEL

Da es sich um einen Datensatz mit ca. 500 Bildern handelt wurde für zwei Knotenstunde trainiert. Das tatsächliche trainieren hat ca. 1,5 Stunden gedauert.

Leitfaden für Modell-Einstellungen:

| Optimize model for... | The model configuration to use. You can train faster, smaller, models when low latency or small package size are important, or slower, larger, models when accuracy is most important. | | | | | | | | | | | | | | | | | | |
|------------------------|---|------------------------|--|-----------------|--------|------------|---------|--------------|---------|--------------|---------|---------------|---------|---------------|----------|----------------|----------|------------------|----------|
| Node hour budget | <p>The maximum time, in compute hours, to spend training the model. More training time generally results in a more accurate model.</p> <p>Note that training can be completed in less than the specified time if the system determines that the model is optimized and additional training would not improve accuracy. You are billed only for the hours actually used.</p> <table border="1"> <thead> <tr> <th colspan="2">Typical training times</th> </tr> </thead> <tbody> <tr> <td>Very small sets</td><td>1 hour</td></tr> <tr> <td>500 images</td><td>2 hours</td></tr> <tr> <td>1,000 images</td><td>3 hours</td></tr> <tr> <td>5,000 images</td><td>6 hours</td></tr> <tr> <td>10,000 images</td><td>7 hours</td></tr> <tr> <td>50,000 images</td><td>11 hours</td></tr> <tr> <td>100,000 images</td><td>13 hours</td></tr> <tr> <td>1,000,000 images</td><td>18 hours</td></tr> </tbody> </table> | Typical training times | | Very small sets | 1 hour | 500 images | 2 hours | 1,000 images | 3 hours | 5,000 images | 6 hours | 10,000 images | 7 hours | 50,000 images | 11 hours | 100,000 images | 13 hours | 1,000,000 images | 18 hours |
| Typical training times | | | | | | | | | | | | | | | | | | | |
| Very small sets | 1 hour | | | | | | | | | | | | | | | | | | |
| 500 images | 2 hours | | | | | | | | | | | | | | | | | | |
| 1,000 images | 3 hours | | | | | | | | | | | | | | | | | | |
| 5,000 images | 6 hours | | | | | | | | | | | | | | | | | | |
| 10,000 images | 7 hours | | | | | | | | | | | | | | | | | | |
| 50,000 images | 11 hours | | | | | | | | | | | | | | | | | | |
| 100,000 images | 13 hours | | | | | | | | | | | | | | | | | | |
| 1,000,000 images | 18 hours | | | | | | | | | | | | | | | | | | |

Kosten sind ~3,50\$/Knotenstunde.

Weitere Infos über die Kosten: [Pricing | Vertex AI](#)

Man kann die Bilder selbst als Testdata oder Trainingsdata labeln, wenn man dies nicht macht, werden von AutoML automatisch für jedes Label Bilder "beiseite gelegt", die dann danach gleich für das Testen und für die Evaluierung des Models verwendet werden. In diesem Fall sieht die Verteilung von Train- und Evaluierungs/Testdaten folgendermaßen aus:

Label stats

Unlabelled images aren't used. Your data set will be automatically split into [Train](#), [Validation](#) and [Test sets](#).

Ideally, each label should have at least **10 images**. Fewer images often result in inaccurate precision and recall. You must also have at least **8, 1, 1 images** each assigned to your Train, Validation and Test sets.

| Labels | Images | Train | Validation | Test |
|-------------|--------|-------|------------|------|
| GERING | 197 | 157 | 20 | 20 |
| HOCH | 73 | 59 | 7 | 7 |
| MITTEL | 130 | 104 | 13 | 13 |
| SEHR_GERING | 39 | 31 | 4 | 4 |

d) Modell trainieren lassen

The screenshot shows the Google Cloud Platform Vision API interface. On the left, there's a sidebar with 'Vision' selected. The main area shows a dataset named 'test123'. Below it, there are tabs for 'IMPORT', 'IMAGES', 'TRAIN', 'EVALUATE', and 'TEST & USE'. The 'TRAIN' tab is currently active. A modal window titled 'Models' is open, showing a progress bar for training a model named 'test123_191122'. The progress bar is nearly complete. There's some text in the modal about training time and infrastructure setup.

3) Testen und Evaluation

Wenn das Modell fertig trainiert wurde, wird man über E-Mail benachrichtigt. Im **TRAIN** Tab sieht man nun eine Zusammenfassung der Evaluierung des Modells und hat auch die Möglichkeit das Modell weiter zu trainieren bzw. ein neues Modell zu trainieren.

Im **EVALUATE** Tab werden die Daten über die Performance des Modells genau angezeigt.

| True Label | MITTEL | SEHR_GERING | GERING | HOCH |
|-------------|--------|-------------|--------|------|
| MITTEL | 46% | - | 54% | - |
| SEHR_GERING | - | 50% | 50% | - |
| GERING | - | - | 100% | - |
| HOCH | 14% | - | 57% | 29% |

Dabei wird die Performance des Modells durch zwei Parameter gemessen: der (a) *precision* und dem (b) *recall*

- (a) Im Zusammenhang mit der Bildklassifizierung ist die *precision* das Verhältnis zwischen der Anzahl der Bilder, die richtig beschriftet wurden, und der Anzahl der Bilder, die das Modell bei dem gewählten Schwellenwert insgesamt beschriftet hat. Wenn ein Modell eine hohe

Präzision aufweist, vergibt es seltener falsche Kennzeichnungen (weniger falsch-positive Ergebnisse).

- (b) Der *recall* ist das Verhältnis zwischen der Anzahl der Bilder, die richtig beschriftet wurden, und der Anzahl der Bilder mit Inhalten, die das Modell hätte beschriften können. Wenn ein Modell einen hohen Recall hat, kann es seltener eine Kennzeichnung nicht zuordnen (weniger falsch-negative Ergebnisse).

Mittels dem Verschieben des *Confidence Threshold* kann man sehen wie sich verschiedene Schwellenwerte auf die Performance des Modells auswirken. Der *Confidence Threshold* gibt die Mindestkonfidenz an, die das Modell haben soll, damit es einem Bild eine Bezeichnung zuweist. Ob man dabei das Modell für *precision* oder *recall* optimiert hängt von der Nutzung ab. Unter [AutoML Vision Beginner's guide | Google Cloud](#) und [Inclusive ML | Google Cloud](#) finden sich Informationen zu den Einstellungen von *precision* und *recall*

4) Test und Verwendung

Das Modell kann danach als TensorFlow.js exportiert werden. Damit kann man das Modell dann im Browser verwenden.

[Tensorflow JS Tutorial](#)

a) **TensorFlow.js** als Möglichkeit auswählen

The screenshot shows the Google Cloud Platform Vision API interface. The 'TEST & USE' tab is active. Under the 'Use your model' section, the 'TensorFlow.js' option is highlighted with a red box. It is described as: 'Export your model as a TensorFlow.js package to run your model in the browser and in Node.js.'

- b) Gewünschten Ordner im Google Storage, wo das Modell gespeichert werden soll, auswählen (sollte im gleichen Bucket wie Bilder, etc. sein)

Export TensorFlow.js package

You can use your model in a web browser or on node.js using the [TensorFlow.js](#) JavaScript library.

1. Export your model as a TensorFlow.js package.
Destination folder on Cloud Storage **BROWSE**

EXPORT **OPEN IN GCS**

2. After your model finishes exporting, you can copy your package to your computer using this command:
`$ gsutil cp -r gs://rock-ai-us-central1/test123/ ./`
3. Follow the quickstart to learn how to implement your model. [Image Classification quickstart](#) [Image Object Detection quickstart](#)

- c) Das Modell exportieren (Anmerkung: ein einfacher Klick auf Export genügt, es wird keine extra Rückmeldung von der Seite gegeben, dass es exportiert wurde)

Export TensorFlow.js package

You can use your model in a web browser or on node.js using the **TensorFlow.js** JavaScript library.

1. Export your model as a TensorFlow.js package.

Destination folder on Cloud Storage
 BROWSE

EXPORT

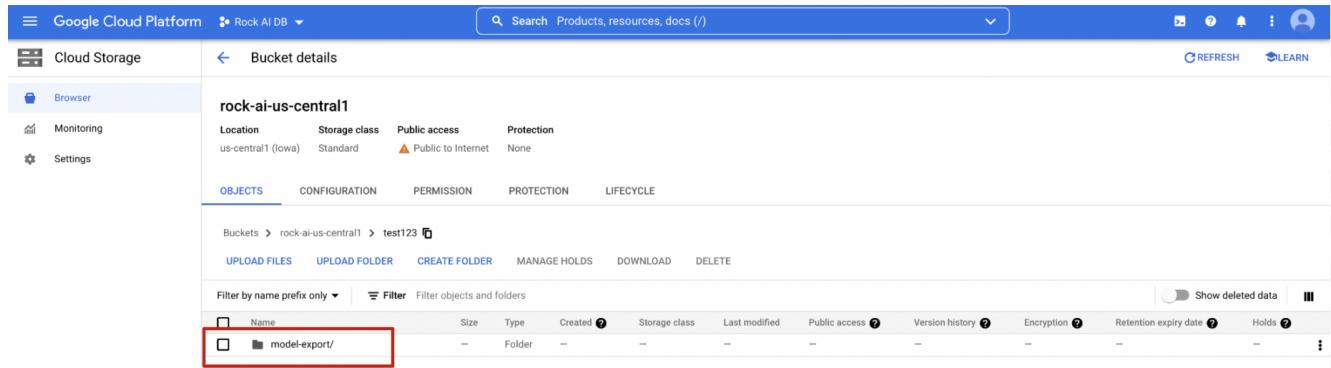
OPEN IN GCS

2. After your model finishes exporting, you can copy your package to your computer using this command:

```
$ gsutil cp -r gs://rock-ai-us-central1/test123/ ./
```

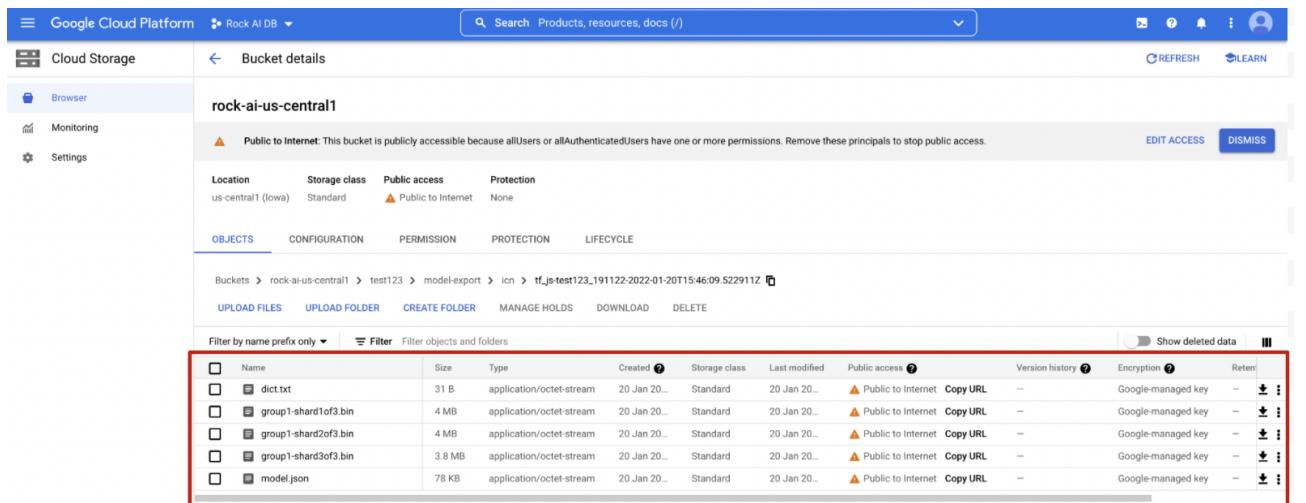
3. Follow the quickstart to learn how to implement your model. [Image Classification quickstart](#) [Image Object Detection quickstart](#)

- d) Nach dem Exportieren wurde in dem ausgewählten Ordner der Unterordner **model-export/** erstellt.



The screenshot shows the 'Bucket details' page for 'rock-ai-us-central1'. Under the 'OBJECTS' tab, there is one entry: 'model-export/' which is a folder. This folder is highlighted with a red box.

In dem Ordner befinden sich dann folgende Dateien



The screenshot shows the 'Bucket details' page for 'rock-ai-us-central1'. Under the 'OBJECTS' tab, there are five entries in the 'model-export/' folder, each highlighted with a red box:

| Name | Type | Created | Storage class | Last modified | Public access | Version history | Encryption | Retention expiry date | Holds |
|----------------------|--------------------------|--------------|---------------|---------------|--------------------|-----------------|------------|-----------------------|-------|
| dict.txt | application/octet-stream | 20 Jan 20... | Standard | 20 Jan 20... | Public to Internet | Copy URL | — | Google-managed key | — |
| group1-shard0f3.bin | application/octet-stream | 20 Jan 20... | Standard | 20 Jan 20... | Public to Internet | Copy URL | — | Google-managed key | — |
| group1-shard2of3.bin | application/octet-stream | 20 Jan 20... | Standard | 20 Jan 20... | Public to Internet | Copy URL | — | Google-managed key | — |
| group1-shard3of3.bin | application/octet-stream | 20 Jan 20... | Standard | 20 Jan 20... | Public to Internet | Copy URL | — | Google-managed key | — |
| model.json | application/octet-stream | 20 Jan 20... | Standard | 20 Jan 20... | Public to Internet | Copy URL | — | Google-managed key | — |

- e) Wenn sich diese exportierten Dateien im Google Cloud Storage Ordner befinden, werden diese in ein lokales Verzeichnis kopiert. Dafür muss **cloud-sdk** installiert sein: [Quickstart: Getting started with Cloud SDK | Cloud SDK Documentation | Google Cloud](#)
 Weiters wird Python 3.7 benötigt: [Python Release Python 3.7.0](#)
 Und Tensorflow: [Installation | TensorFlow Hub](#)

Wenn cloud-sdk installiert wurde, geschieht das Kopieren mit dem folgenden Kommando:

```
gsutil cp -r gs://[path in cloud-storage]/model-export/ {local-folder}
```

In unserem Fall also:

```
gsutil cp -r gs://rock-ai-us-central1/test123/model-export/ {local-folder}
```

- f) Nachdem die Dateien lokal gespeichert wurden, wird folgendes **index.html** File im Ordner, wo diese Dateien lokal gespeichert wurden erstellt:



```
index.html
x
<!DOCTYPE html>
<html lang="en">
<script src="https://unpkg.com/@tensorflow/tfjs"></script>
<script src="https://unpkg.com/@tensorflow/tfjs-automl"></script>

<script>
async function run() {
  const model = await tf.automl.loadImageClassification('model.json');
  const image = document.getElementById('image00_00');
  const predictions = await model.classify(image);
  console.log(predictions);
  // Show the resulting object on the page.
  const pre = document.createElement('pre');
  pre.textContent = JSON.stringify(predictions, null, 2);
  document.body.append(pre);
}
run();
</script>
```

Code:

```
<!DOCTYPE html>
<html lang="en">
<script src="https://unpkg.com/@tensorflow/tfjs"></script>
<script src="https://unpkg.com/@tensorflow/tfjs-automl"></script>

<script>
async function run() {
  const model = await tf.automl.loadImageClassification('model.json');
  const image = document.getElementById('image00_00');
  const predictions = await model.classify(image);
  console.log(predictions);
  // Show the resulting object on the page.
  const pre = document.createElement('pre');
  pre.textContent = JSON.stringify(predictions, null, 2);
```

```

        document.body.append(pre);
    }
    run();
</script>

```

Wobei der img src Link durch den Pfad zu dem Bild, das klassifiziert werden soll ersetzt werden muss. Dieses Bild muss einen öffentlich zugänglichen Google Cloud Storage-Pfad besitzen.

- g) Im Terminal startet man nun einen einfachen HTTP-Server an Port 8000 aus dem Verzeichnis mit der Datei index.html.

Dafür gibt es zwei Möglichkeiten, die erste ist dies mit Python durchzuführen:

- 1) **Python:** `python -m SimpleHTTPServer 8000`

A terminal window showing the command `python -m SimpleHTTPServer 8000` being run. The output shows "Serving HTTP on 0.0.0.0 port 8000 ...".

Die zweite ist dies mit Node zu machen: [Download | Node.js](#)

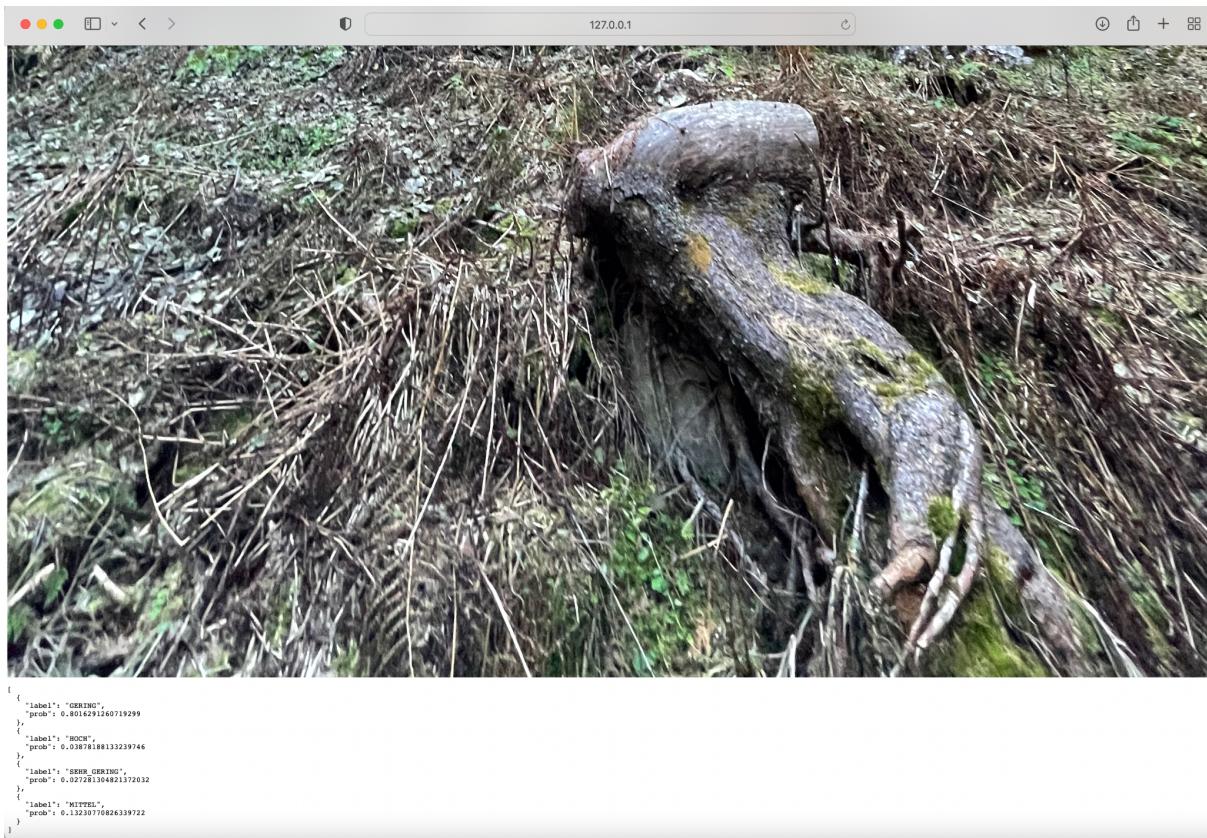
- 2) **Node.js:** `http-server -p 8000`

A terminal window showing the command `http-server -p 8000` being run. The output shows "Starting up http-server, serving ./". It then lists various server settings including CORS: disabled, Cache: 3600 seconds, Connection Timeout: 120 seconds, Directory Listings: visible, AutoIndex: visible, Serve GZIP Files: false, Serve Brotli Files: false, and Default File Extension: none. It also shows the available URLs: `http://127.0.0.1:8000` and `http://192.168.0.4:8000`, and a message to hit CTRL-C to stop the server.

- h) Im Browser öffnet man dann für

- 1) <http://localhost:8000/> und für
- 2) <http://127.0.0.1:8000/> oder <http://192.168.0.4:8000/>

- i) Im Browser erscheint dann das Bild mit den automatisch vorhergesagten Labels (klein links-unten im Bild)



Einrichten eines Modells für Online-Nutzung

Es gibt auch die Möglichkeit, das Modell Cloud-basiert zu nutzen. Die Schritte für das Einrichten so eines Modells sind dieselben wie bereits beschrieben. Der Unterschied dabei ist, dass anstatt **Edge**, **Cloud hosted** ausgewählt wird.

Train new model

1 Define your model

Model name *
test123_20220120090630

Cloud hosted

Host your model on Google Cloud for online predictions

Edge

Download your model for offline/mobile use

CONTINUE

2 Set a node hour budget

START TRAINING

CANCEL

Es gibt auch die Möglichkeit das Modell, das zuerst im Edge-Modus definiert wurde nach dem Training direkt im **TEST & USE** Tab in die Cloud bereitzustellen

The screenshot shows the Google Cloud Platform Vision interface. The top navigation bar includes 'Google Cloud Platform', 'Rock AI DB', a search bar, and user icons. The left sidebar has 'Vision' selected, followed by 'Dashboard', 'Data sets' (which is highlighted in blue), and 'Models'. The main content area is titled 'TEST & USE' with tabs for 'IMPORT', 'IMAGES', 'TRAIN', 'EVALUATE', and 'TEST & USE'. A dropdown menu under 'Model' shows 'test123_191122'. Below this, a red box surrounds a callout message: 'To use online prediction, deploy your model to the cloud. Deployed model charges are per hour and number of machines used.' with a link to 'Pricing guide'. Another message below it says: 'Notice for beta users: The v1beta1 API endpoint is scheduled for deletion after GA release. If your beta models have not been redeployed since October 17, 2019, please do so now to avoid interruption when the old service is shut down.' A 'DEPLOY MODEL' button is located at the bottom right of this section. The 'Use your model' section contains five options: 'TF Lite', 'TensorFlow.js', 'Core ML', 'Container', and 'Coral'.

Wenn das Modell in die Cloud bereitgestellt wurde kann im TEST & USE Tab direkt ein Bild/mehrere Bilder hochgeladen werden, die dort dann gelabelt werden.

The screenshot shows the Google Cloud Platform Vision interface with a deployed model named 'rockAI_singleLabel2'. The 'TEST & USE' tab is active. A green checkmark message says: 'Your model is deployed and is available for online prediction requests.' with a link to 'Learn more'. Below it is the same beta notice as the previous screenshot. The 'Test your model' section features a large red box around the 'UPLOAD IMAGES' button. A red arrow points from this button to a predicted image of a rock face. The predicted image shows a close-up of a rock surface with a small waterfall, labeled 'Predictions' with '1 object' and 'MITTEL' with a confidence score of '0.961608350.96'.

Trainierte Modelle und Ergebnisse

Im folgenden werden die Evaluationsergebnisse der trainierten Modelle (in chronologischer Reihenfolge) beschrieben.

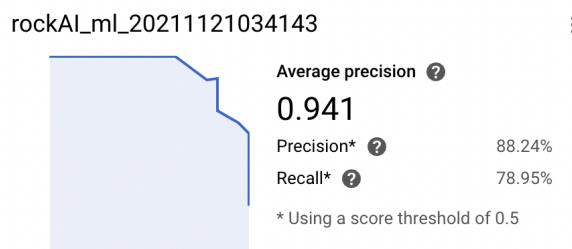
Anmerkung: der Confidence Threshold wurde bei allen Evaluierungen auf 0.5 gestellt.

1. Modell: rockAI_ml

Erstellt am 21.11.2021, Single-Label Classification mit 193 Bildern

| | | | | | | | |
|---|-----------------------------|-----|-----|-----------------------|-------------------------|----|---|
| <input checked="" type="checkbox"/> rockAI_ml | Single-Label Classification | 193 | 193 | 21 Nov 2021, 15:40:10 | Success: Training model | No | ⋮ |
|---|-----------------------------|-----|-----|-----------------------|-------------------------|----|---|

Evaluation:



| | |
|----------------------------|------------------------|
| Model ID ? | ICN8008013664953040896 |
| Created | 21 Nov 2021, 15:43:16 |
| Base model | None |
| Data | 193 images |
| Model type | Mobile Best Trade-Off |
| Train cost | 1 node hour |
| Deployment state | Not deployed |

IMPORT IMAGES TRAIN EVALUATE TEST & USE Single-Label Classification

Model Confidence threshold

| All labels | |
|-------------|---------|
| All labels | 0.94143 |
| GERING | 0.97159 |
| HOCH | 1 |
| MITTEL | 0.885 |
| SEHR_GERING | 1 |

Total images 174 Test items 19 Precision [?](#) 88.24% Recall [?](#) 78.95%

Use the slider to see which confidence threshold works best for your model on the precision-recall tradeoff curve. [Learn more about these metrics and graphs.](#)

Precision Recall Confidence

Confusion matrix

This table shows how often the model classified each label correctly (in blue), and which labels were most often confused for that label (in grey). You can download the entire confusion matrix as a CSV file.

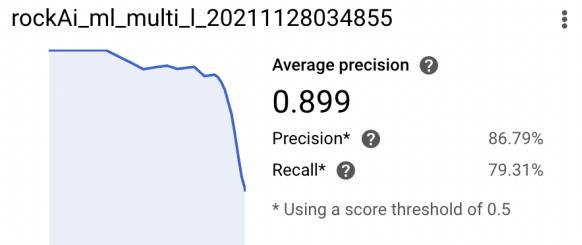
| True Label | Predicted Label | | | |
|-------------|-----------------|------|-------------|--------|
| | GERING | HOCH | SEHR_GERING | MITTEL |
| GERING | 91% | 9% | - | - |
| HOCH | - | 100% | - | - |
| SEHR_GERING | - | - | 100% | - |
| MITTEL | 40% | - | - | 60% |

2. Modell: rockAi_ml_multi_label

Erstellt am 28.11.2021, Multi-Label Classification mit 189 Bildern

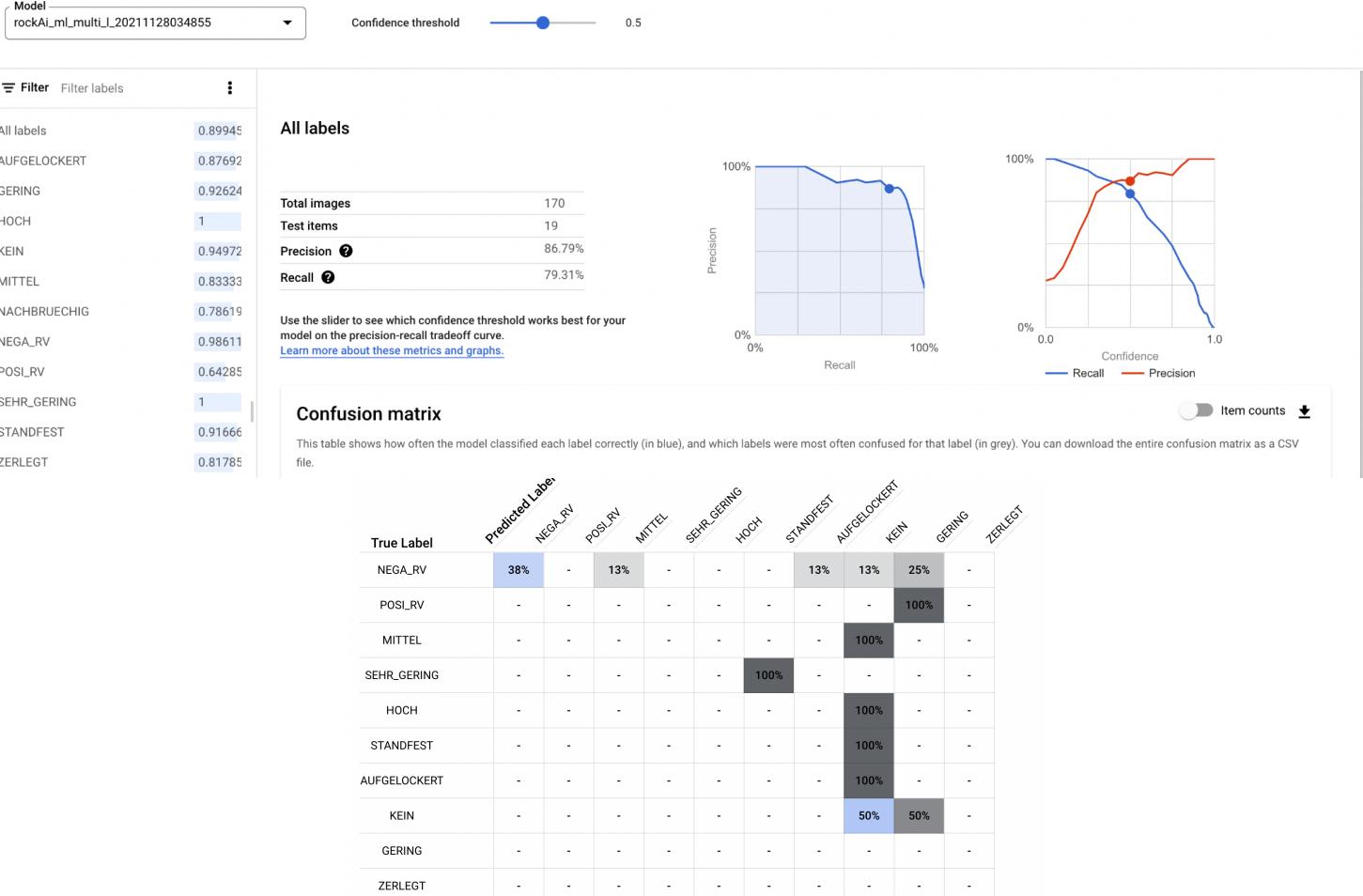
| | | | | | | | |
|---|----------------------------|-----|-----|-----------------------|-------------------------|----|---|
| <input checked="" type="checkbox"/> rockAi_ml_multi_label | Multi-Label Classification | 189 | 189 | 28 Nov 2021, 15:28:42 | Success: Training model | No | ⋮ |
| ICN3757212101773361152 | | | | | | | |

Evaluation:



| | |
|-------------------------|------------------------|
| Model ID ? | ICN6294112531761856512 |
| Created | 28 Nov 2021, 15:49:57 |
| Base model | None |
| Data | 189 images |
| Model type | Mobile Best Trade-Off |
| Train cost | 0.714 node hours |
| Deployment state | Not deployed |

| IMPORT | IMAGES | TRAIN | EVALUATE | TEST & USE | Metric |
|--------|--------|-------|----------|------------|----------------------------|
| | | | | | Multi-Label Classification |



Anmerkungen zu 1. und 2. Modell:

Die Average Precisions sind mit Vorsicht zu genießen, da nur sehr wenige Bilder für das Trainieren zur Verfügung standen. (Normalerweise werden Datensätze mit 1000+ Bildern verwendet)

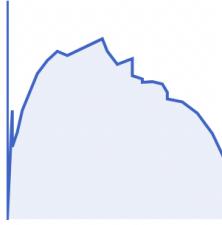
3. Modell: rockAi_singleLabels2

Erstellt am 11.01.2022, Single-Label Classification mit 439 Bildern

| | | | | | | | |
|--|-----------------------------|-----|-----|-----------------------|-------------------------|----|---|
| <input checked="" type="checkbox"/> rockAi_singleLabels2 | Single-Label Classification | 439 | 439 | 11 Jan 2022, 13:52:54 | Success: Training model | No | ⋮ |
|--|-----------------------------|-----|-----|-----------------------|-------------------------|----|---|

Modell A: Edge-basiert

Evaluation: rockAi_singleLabe_220111 ⋮



Average precision ⓘ
0.63
 Precision* ⓘ 65.79%
 Recall* ⓘ 56.82%
* Using a score threshold of 0.5

| | |
|------------------|------------------------|
| Model ID | ICN3213157805431193600 |
| Created | 11 Jan 2022, 13:54:05 |
| Base model | None |
| Data | 439 images |
| Model type | Mobile Best Trade-Off |
| Train cost | 1.043 node hours |
| Deployment state | Not deployed |

IMPORT IMAGES TRAIN EVALUATE TEST & USE Single-Label Classification

Model: rockAi_singleLabe_220111

Confidence threshold: 0.5

All labels

| | |
|-------------|---------|
| All labels | 0.63031 |
| GERING | 0.62445 |
| HOCH | 0.72733 |
| MITTEL | 0.73153 |
| SEHR_GERING | 0.80882 |

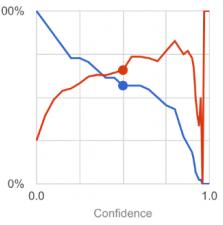
Use the slider to see which confidence threshold works best for your model on the precision-recall tradeoff curve.
[Learn more about these metrics and graphs.](#)

Precision



Recall

Precision



Confidence

Item counts

Confusion matrix

This table shows how often the model classified each label correctly (in blue), and which labels were most often confused for that label (in grey). You can download the entire confusion matrix as a CSV file.

| True Label | Predicted Label | | | |
|-------------|-----------------|--------|------|--------|
| | SEHR_GERING | MITTEL | HOCH | GERING |
| SEHR_GERING | 75% | - | - | 25% |
| MITTEL | 8% | 54% | - | 38% |
| HOCH | - | 14% | 29% | 57% |
| GERING | 5% | 15% | - | 80% |

Modell B: Cloud-basiert

Evaluation:

rockAi_singleLabel_cloudPred

⋮

Average precision [?](#)

0.859

Precision* [?](#)

79.55%

Recall* [?](#)

79.55%

* Using a score threshold of 0.5

| | |
|----------------------------|-----------------------|
| Model ID ? | ICN480669100268322816 |
| Created | 11 Jan 2022, 15:56:21 |
| Base model | None |
| Data | 439 images |
| Model type | Cloud |
| Train cost | 10 node hours |
| Deployment state | Deployed |

IMPORT IMAGES TRAIN EVALUATE TEST & USE Single-Label Classification

Model rockAi_singleLabel_cloudPred Confidence threshold 0.5

| Filter | Filter labels | ⋮ |
|-------------|---------------|---|
| All labels | 0.85927 | |
| GERING | 0.90116 | |
| HOCH | 0.80221 | |
| MITTEL | 0.89663 | |
| SEHR_GERING | 1 | |

Total images 395
Test items 44
Precision [?](#) 79.55%
Recall [?](#) 79.55%

Use the slider to see which confidence threshold works best for your model on the precision-recall tradeoff curve.
[Learn more about these metrics and graphs.](#)

Precision-Recall curve (left): A blue line plot showing Precision (Y-axis, 0% to 100%) versus Recall (X-axis, 0% to 100%). A single point is plotted at (Recall: 0.5, Precision: 0.859).

ROC curve (right): A red line plot showing Precision (Y-axis, 0% to 100%) versus Confidence (X-axis, 0.0 to 1.0). A single point is plotted at (Confidence: 0.5, Precision: 0.859).

Confusion matrix (bottom): A table showing the number of correctly classified and misclassified instances for each label pair.

| True Label | Predicted Label | | | | |
|-------------|-----------------|--------|------|--------|--|
| | SEHR_GERING | MITTEL | HOCH | GERING | |
| SEHR_GERING | 100% | - | - | - | |
| MITTEL | - | 85% | - | 15% | |
| HOCH | - | - | 71% | 29% | |
| GERING | 5% | 15% | 5% | 75% | |

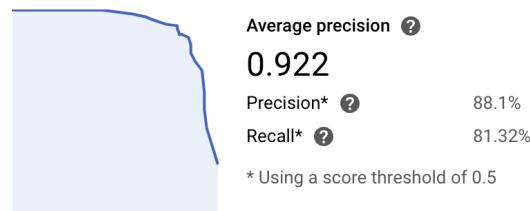
4. Modell: rockAI_singleLabel_augmented

Erstellt am 21.01.2022, Single-Label Classification mit 894 Bildern (Bilder vom Modell davor wurden modifiziert, damit Menge an Daten vergrößert wird)

| | | | | | | | |
|---|-----------------------------|-----|-----|-----------------------|-------------------------|----|---|
| <input checked="" type="checkbox"/> rockAI_singleLabels_augmented | Single-Label Classification | 894 | 894 | 21 Jan 2022, 19:30:19 | Success: Training model | No | ⋮ |
|---|-----------------------------|-----|-----|-----------------------|-------------------------|----|---|

Evaluation:

rockAI_singleLabel_augmented ⋮



| | |
|-------------------------|------------------------|
| Model ID ? | ICN5184802056359116800 |
| Created | 21 Jan 2022, 19:31:09 |
| Base model | None |
| Data | 894 images |
| Model type | Mobile Best Trade-Off |
| Train cost | 2.062 node hours |
| Deployment state | Not deployed |

IMPORT IMAGES TRAIN EVALUATE TEST & USE Single-Label Classification

Model: rockAI_singleLabel_augmented Confidence threshold: 0.5

| All labels | Precision | Recall |
|-------------|-----------|---------|
| All labels | 0.92171 | 0.92224 |
| GERING | 0.99224 | 0.95689 |
| HOCH | 0.95689 | 0.95017 |
| MITTEL | 0.95017 | 0.91847 |
| SEHR_GERING | 0.91847 | 0.92171 |

All labels

Total images: 803
Test items: 91
Precision: 88.1%
Recall: 81.32%

Use the slider to see which confidence threshold works best for your model on the precision-recall tradeoff curve.
Learn more about these metrics and graphs.

Confusion matrix

This table shows how often the model classified each label correctly (in blue), and which labels were most often confused for that label (in grey). You can download the entire confusion matrix as a CSV file.

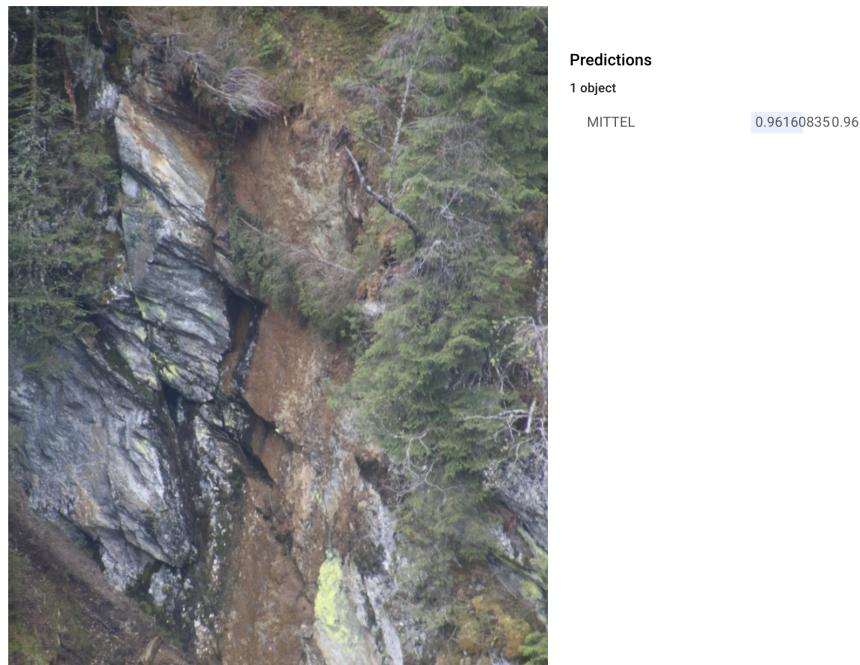
| True Label | Predicted Label | | | |
|-------------|-----------------|--------|------|--------|
| | SEHR_GERING | MITTEL | HOCH | GERING |
| SEHR_GERING | 63% | 13% | - | 25% |
| MITTEL | - | 88% | - | 12% |
| HOCH | - | 11% | 61% | 28% |
| GERING | - | 3% | - | 98% |

Hier sind drei Testungen von dem Cloud-basierten Modell zu sehen:

1. Bild: image00_00:

erwartetes Label: HOCH

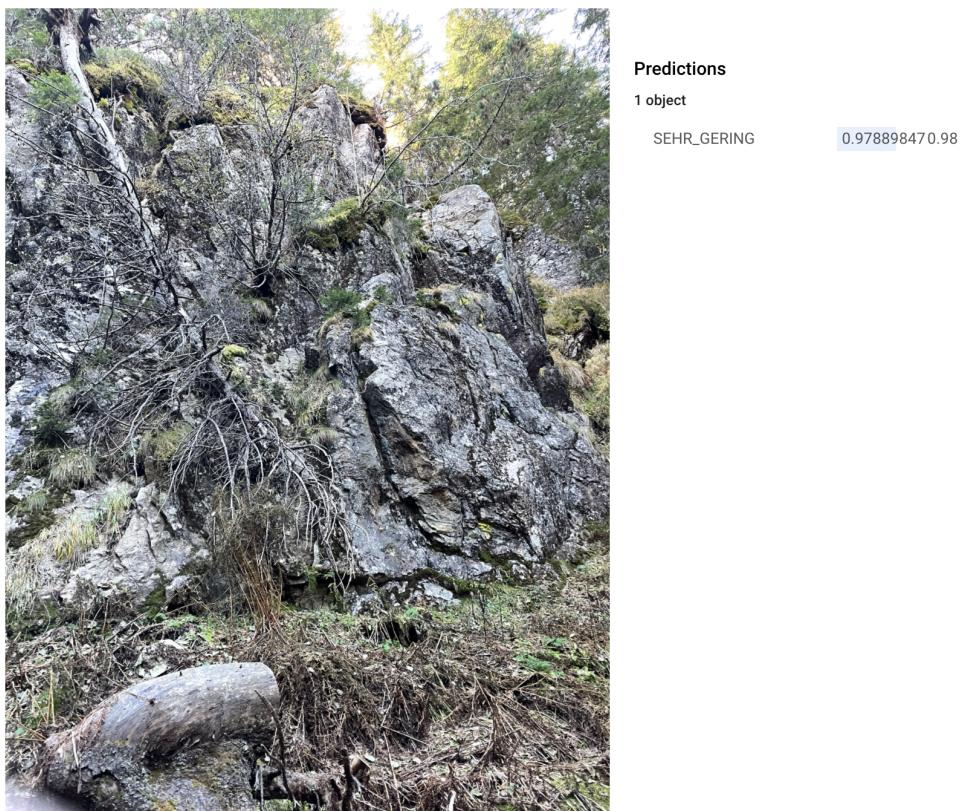
vorhergesagtes Label: MITTEL (96% Wahrscheinlichkeit)



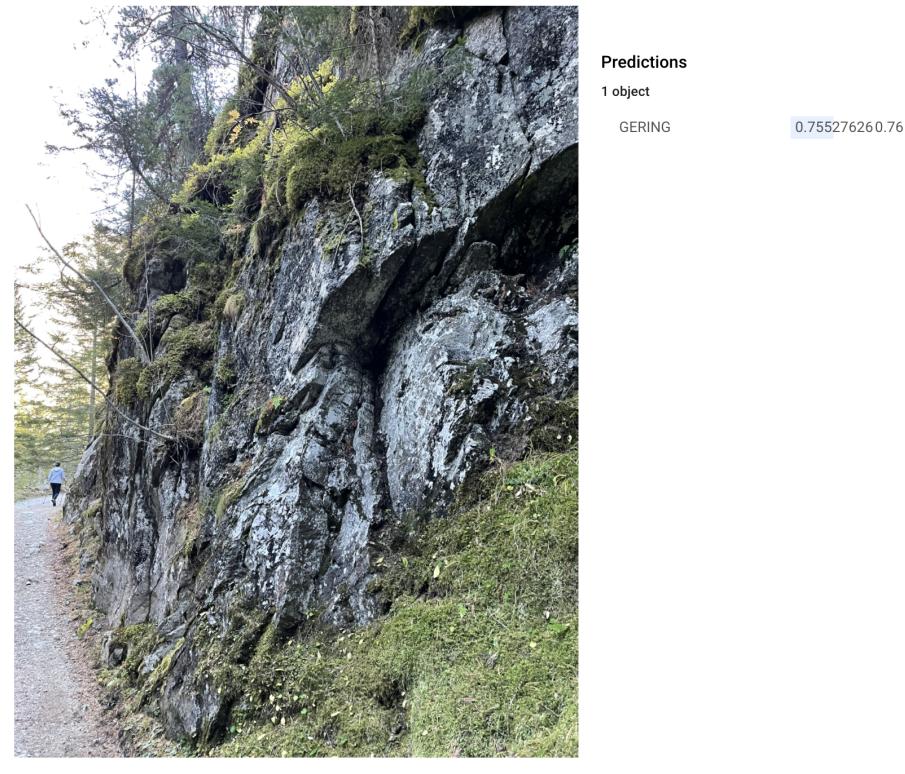
2. Bild: image39_00

erwartetes Label: GERING

vorhergesagtes Label: SEHR GERING (97,8% Wahrscheinlichkeit)



3. Bild: image40_00
erwartetes Label: MITTEL
vorhergesagtes Label: GERING (75% Wahrscheinlichkeit)



Fazit

Alles in allem funktioniert der Service sehr gut. Es gibt auch weitere Tutorials, wie die lokale/cloud-basierte Vorhersage in die App dann eingebaut werden kann. Es ist aber zu bedenken, dass die Anzahl der Bilder natürlich eine große Rolle spielen. Auch funktioniert die Cloud-basierte Vorhersage um einiges besser als die lokale Verwendung des Modells. Bei der lokalen Verwendung gab es manchmal Probleme mit der Anzeige des Bildes und auch zeigen die Predictions der einzelnen Labels bei unterschiedlichen Bildern fast exakt gleiche Ergebnisse. Dieses Problem war darauf zurückzuführen, dass das der Link zu dem zu testendem Bild eine .jpeg o.ä. haben muss. Bei der Cloud-basierten Vorhersage ist aber jeder Zugriff zu bezahlen, siehe [AutoML Vision pricing](#).

Mögliche zukünftige Arbeiten

- beim Testen den öffentlichen Firebase-Link hernehmen und somit Bilder aus der Datenbank testen
- AutoML-Model erstellen/trainieren mit einem größeren Datensatz
zurzeit sind wenig Labels mit der Gefährdungsklasse HOCH, daher sind auch bei diesem Label der Großteil falsch klassifiziert → Image Augmentation verstärkt bei diesem Label betreiben
- Backend für die App erstellen, damit man die Bilder mit dem Machine Learning in der App klassifizieren kann. Dabei an dem [index.html File](#) (Punkt f) orientieren um so ein Backend zu integrieren.
- AutoML-Model auch über die App trainieren können, damit kann der Nutzer direkt die Bilder in die App hochladen und auch von der App aus direkt die Modelle trainieren und auch testen. Das wäre die anschließende Komplettlösung.