Applied Data Science 2021-FS

Projektarbeit - Gruppe 17

Sinthusa Arumugam, Fitim Kuqi & Melvin Thekkenin

zh aw School of Management and Law

Agenda

- Ausgangslage
- Methode & Vorgehensweise
  - Bereitstellung auf GitHub
  - Datenerhebung mittels API
  - Datenbereinigung
  - Datenaufbereitung
- Diskussion & Ergebnisse
- Betrachtung Ethische Fragestellung
- Schlussfolgerungen

Ausgangslage

Hintergrund / Problemstellung

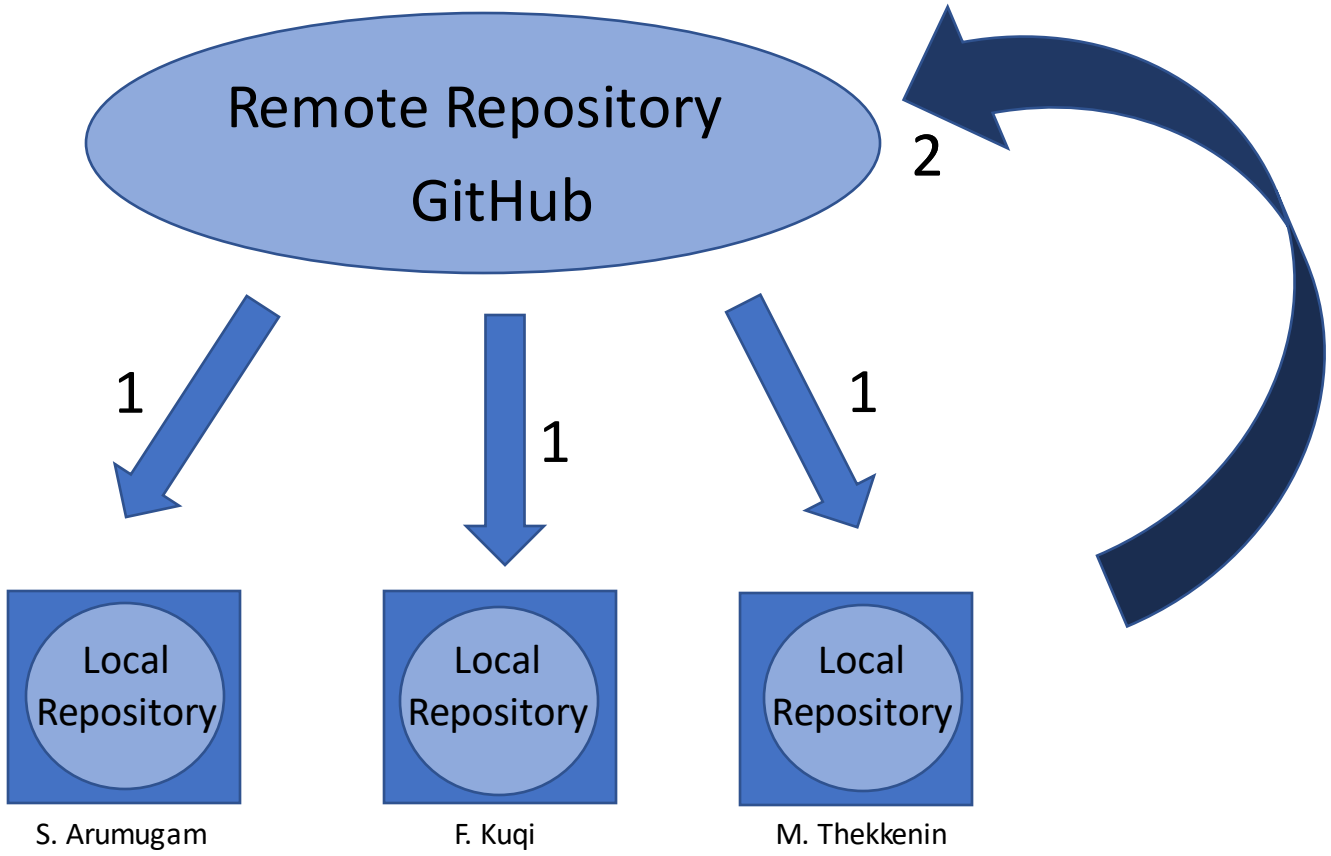
Ziel 1: Mittels "Application Programming Interface" Bilder von Flickr oder Pexel herunterladen.

Ziel 2: Hunde und Katzen mithilfe von Machine Learning Algorithmen korrekt klassifizieren.

Forschungsfrage:

Kann ein ML-Algorithmus mithilfe eines limitierten Trainingsdatensatz korrekt klassifizieren?

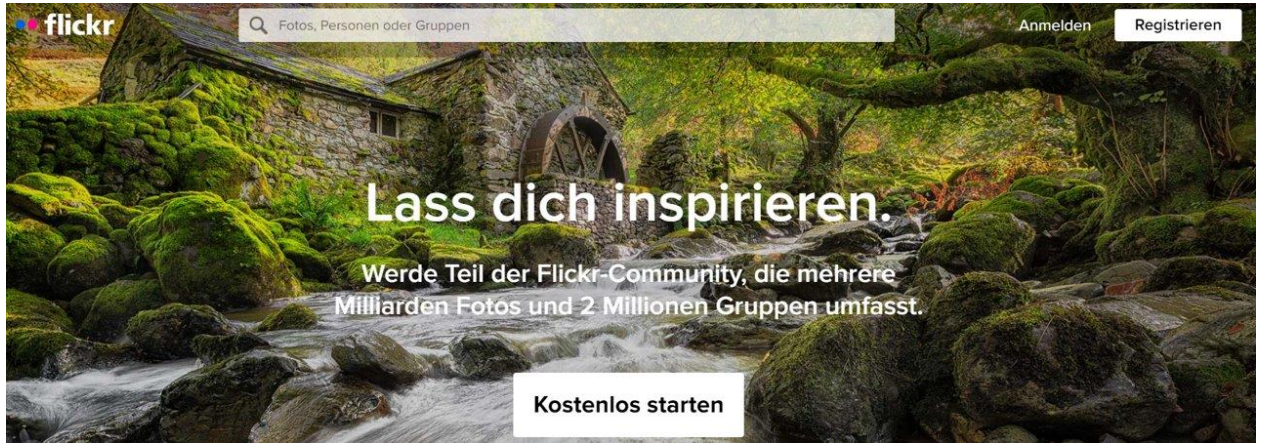
Bereitstellung auf GitHub



API	Update README.md
ML	Delete txt
Pictures	Delete dogs184.jpg

[melvintijo/ADS_Projekt_FMS (github.com)]

Datenerhebung Flickr API

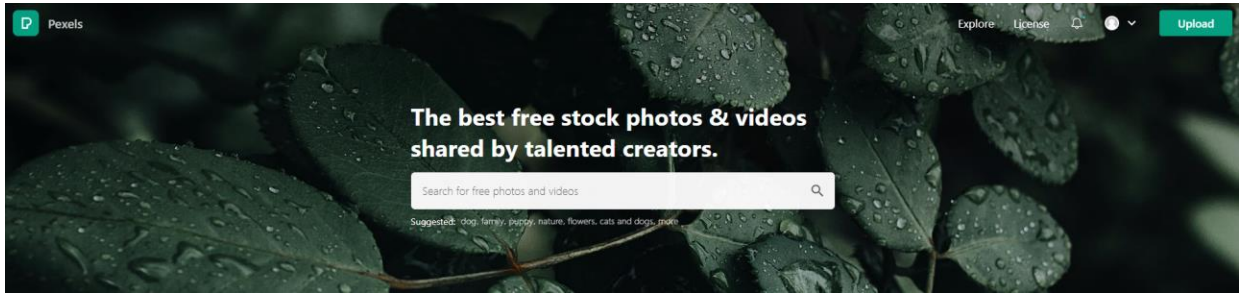


API Flickr

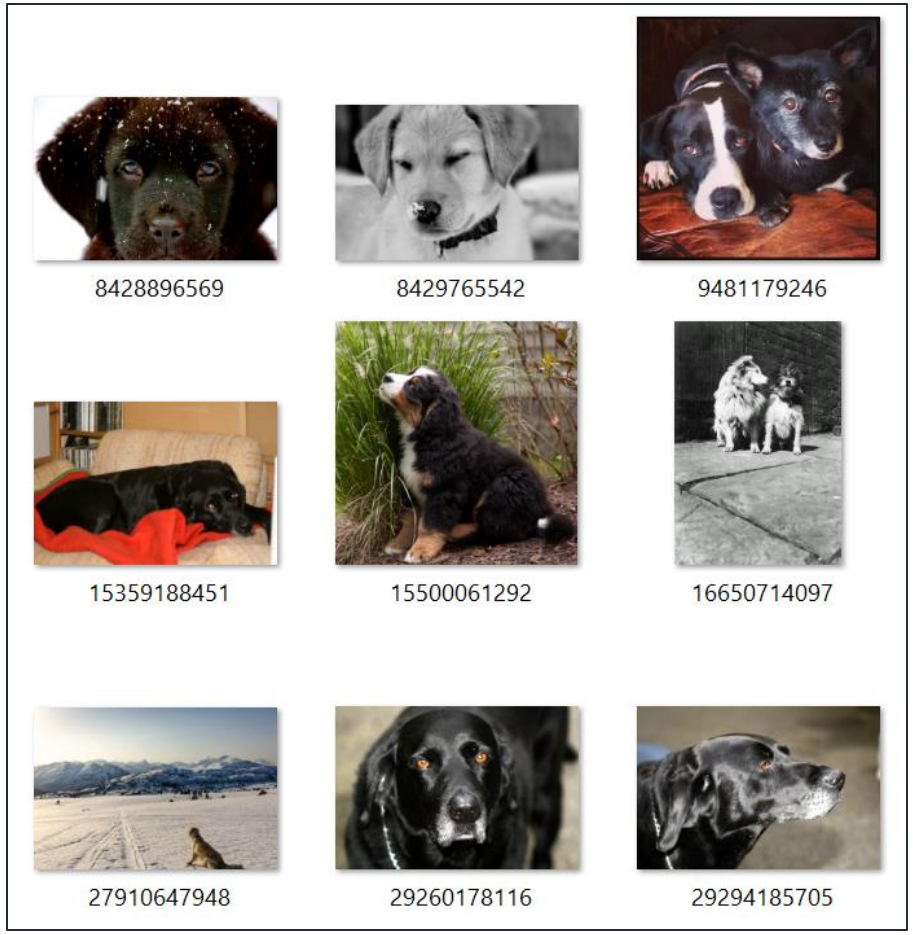
Dog images: python scraper.py --group "https://www.flickr.com/groups/81037160@N00/" --original --max-pages 5

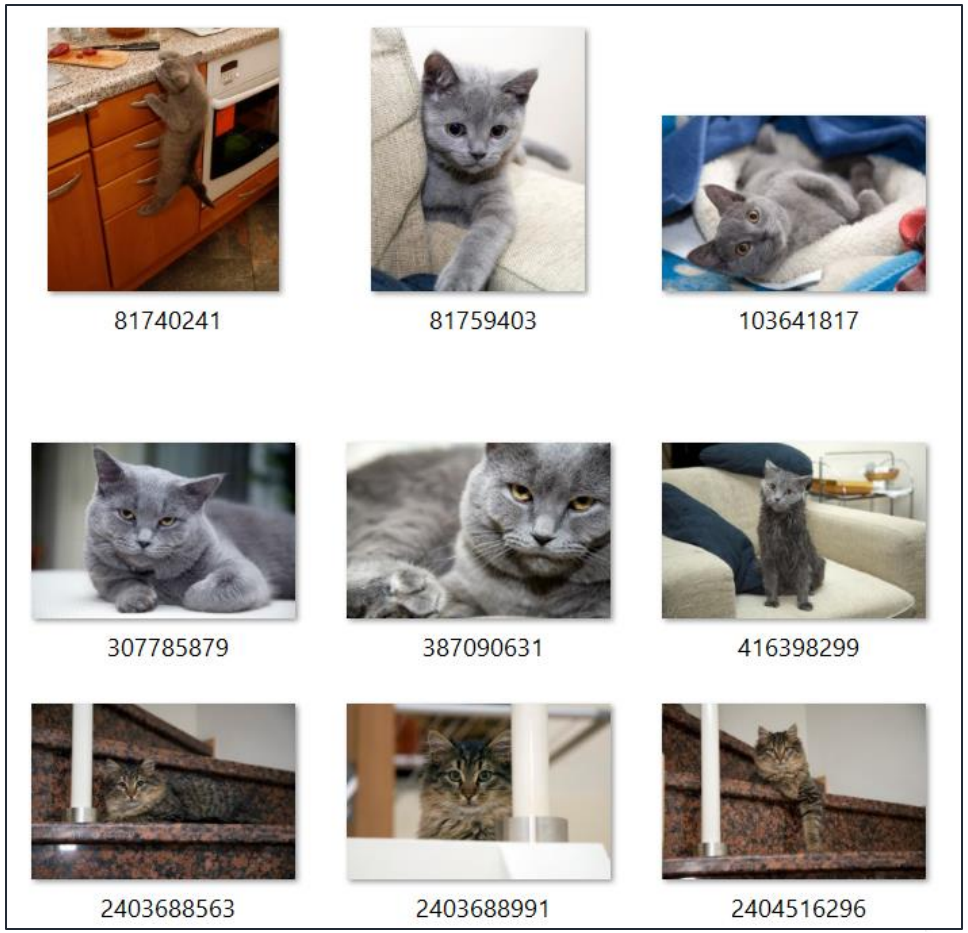
Cat images: python scraper.py --group "https://www.flickr.com/groups/661812@N25/" --original --max-pages 5

Datenerhebung Pexels API



Datenerhebung mittels API





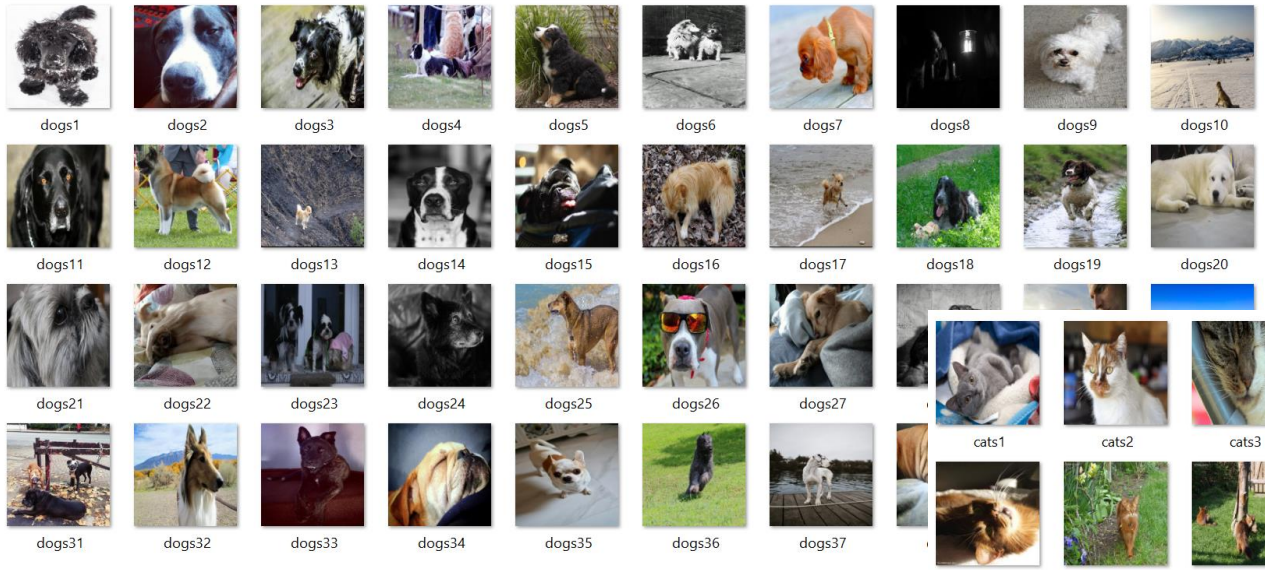
Dogs:

1'782 Bilder

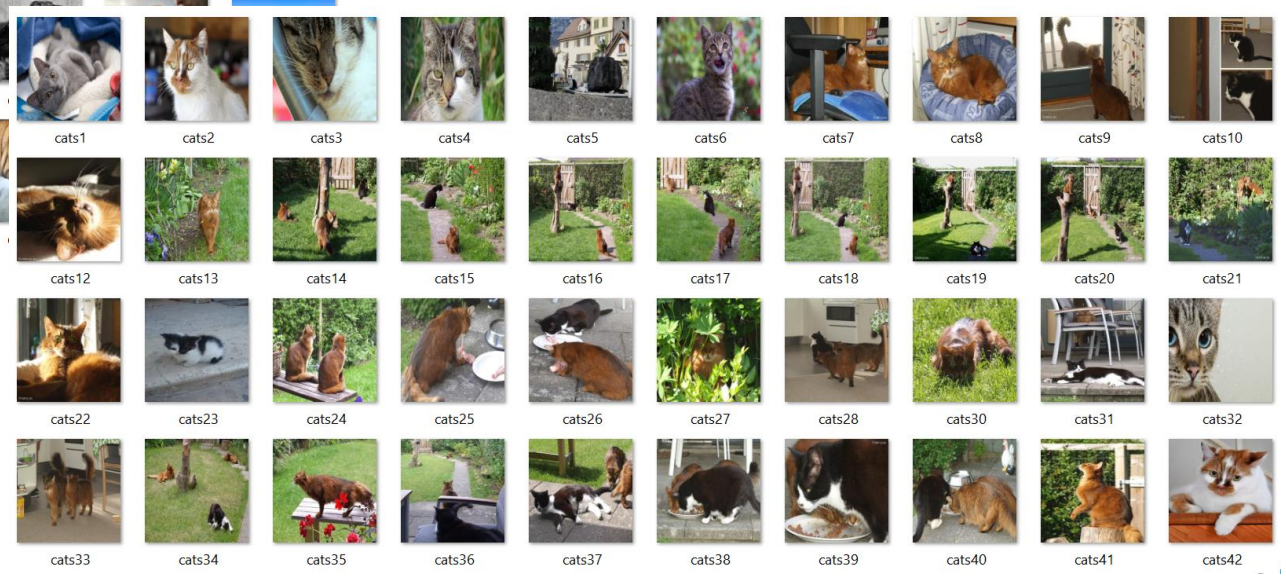
Cats:

1'842 Bilder

Datenbereinigung



Doppelte Bilder gelöscht



Anpassung Grösse: 500 x 500

Dogs: 197 Bilder

Cats: 183 Bilder

Datenaufbereitung - Model

```
model = tf.keras.models.Sequential([
    tf.keras.layers.Conv2D(32, (3,3), activation='relu', input_shape=(IMG_SHAPE, IMG_SHAPE, 3)), # RGB
    tf.keras.layers.MaxPooling2D(2,2),

    tf.keras.layers.Conv2D(64, (3,3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2,2),

    tf.keras.layers.Conv2D(128, (3,3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2,2),

    tf.keras.layers.Conv2D(128, (3,3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2,2),

    tf.keras.layers.Dropout(0.5), # 1/2 of neurons will be turned off randomly
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(256, activation='relu'),

    tf.keras.layers.Dense(2, activation='softmax') #[0, 1] or [1, 0]

    ])


model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])
```

Datenaufbereitung - Model fit

```
history = model.fit_generator(
    train_data_gen,
    steps_per_epoch=int(np.ceil(total_train / float(BATCH_SIZE))),
    epochs=EPOCHS,
    validation_data=val_data_gen,
    validation_steps=int(np.ceil(total_val / float(BATCH_SIZE)))
    )
```

Datenaufbereitung - Overfit reduzieren

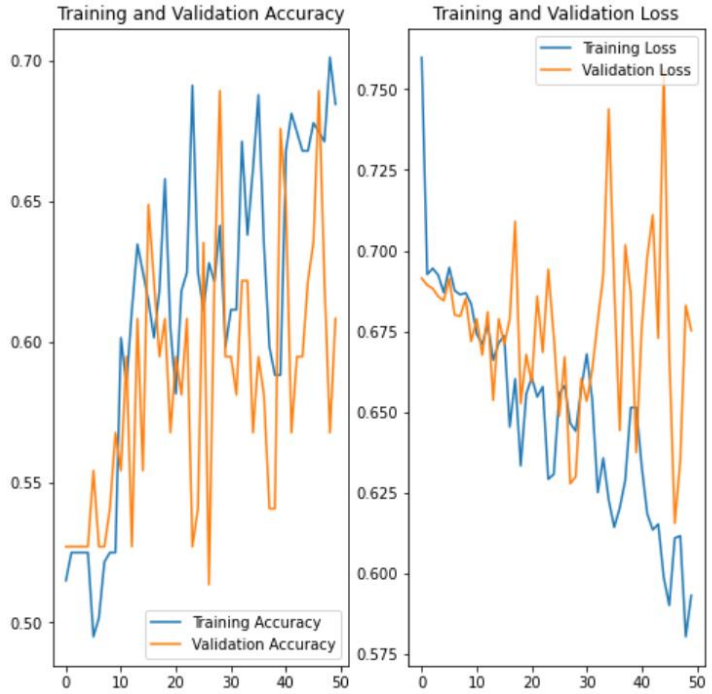
```
train_image_generator = ImageDataGenerator(
    rescale=1./255,
    rotation_range=40,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
    fill_mode='nearest'
    )

validation_image_generator = ImageDataGenerator(
    rescale=1./255)
```

```
tf.keras.layers.Dropout(0.5), # 1/2
```

Datenaufbereitung - Beispiel mit 300 Bildern

```
Epoch 47/50
10/10 [==============================] - 10s 1s/step - loss: 0.6110 - accuracy: 0.6744 - val_loss: 0.6156 - val_accuracy: 0.6892
Epoch 48/50
10/10 [==============================] - 9s 863ms/step - loss: 0.6117 - accuracy: 0.6711 - val_loss: 0.6348 - val_accuracy: 0.6216
Epoch 49/50
10/10 [==============================] - 9s 862ms/step - loss: 0.5805 - accuracy: 0.7010 - val_loss: 0.6831 - val_accuracy: 0.5676
Epoch 50/50
10/10 [==============================] - 9s 872ms/step - loss: 0.5932 - accuracy: 0.6844 - val_loss: 0.6753 - val_accuracy: 0.6081
```



Datenaufbereitung - Transfer Learning – Mobile Net

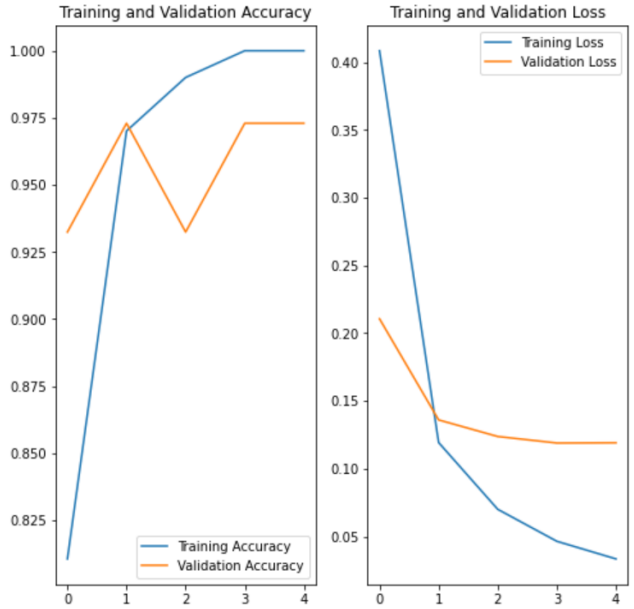
```
# getting MobileNet
URL = "https://tfhub.dev/google/tf2-preview/mobilenet_v2/feature_vector/4"
mobile_net = hub.KerasLayer(URL, input_shape=(IMG_SHAPE, IMG_SHAPE, 3))

mobile_net.trainable = False

model = tf.keras.models.Sequential([
    mobile_net,
    tf.keras.layers.Dense(2, activation='softmax') #[0, 1] or [1, 0]
    ])
```

Diskussion & Ergebnisse

```
Epoch 1/5
61/61 [==============================] - 18s 301ms/step - loss: 0.4085 - accuracy: 0.8106 - val_loss: 0.2107 - val_accuracy: 0.9324
Epoch 2/5
61/61 [==============================] - 12s 198ms/step - loss: 0.1193 - accuracy: 0.9701 - val_loss: 0.1360 - val_accuracy: 0.9730
Epoch 3/5
61/61 [==============================] - 12s 195ms/step - loss: 0.0700 - accuracy: 0.9900 - val_loss: 0.1238 - val_accuracy: 0.9324
Epoch 4/5
61/61 [==============================] - 12s 194ms/step - loss: 0.0464 - accuracy: 1.0000 - val_loss: 0.1190 - val_accuracy: 0.9730
Epoch 5/5
61/61 [==============================] - 12s 193ms/step - loss: 0.0334 - accuracy: 1.0000 - val_loss: 0.1192 - val_accuracy: 0.9730
```



Datenaufbereitung - Transfer Learning – Efficient Net

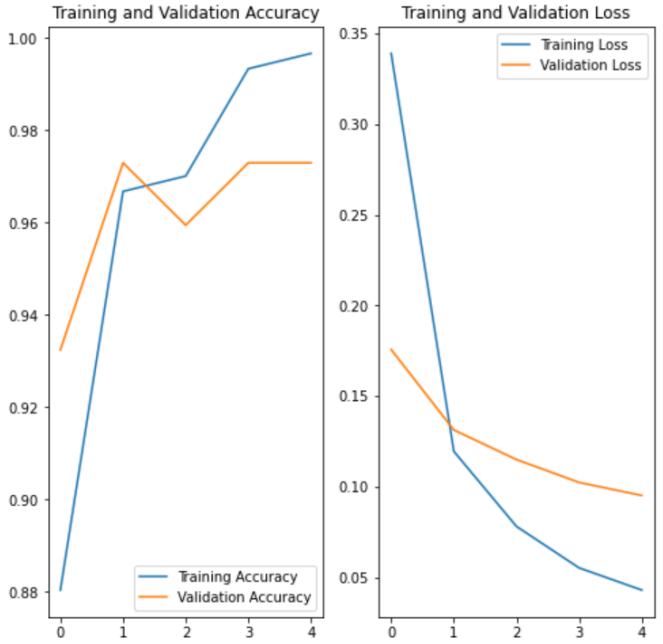
```
# getting Efficientnet
URL = "https://tfhub.dev/tensorflow/efficientnet/lite0/feature-vector/2"
mobile_net = hub.KerasLayer(URL, input_shape=(IMG_SHAPE, IMG_SHAPE, 3))

mobile_net.trainable = False

model = tf.keras.models.Sequential([
    mobile_net,
    tf.keras.layers.Dense(2, activation='softmax') #[0, 1] or [1, 0]
    ])
```

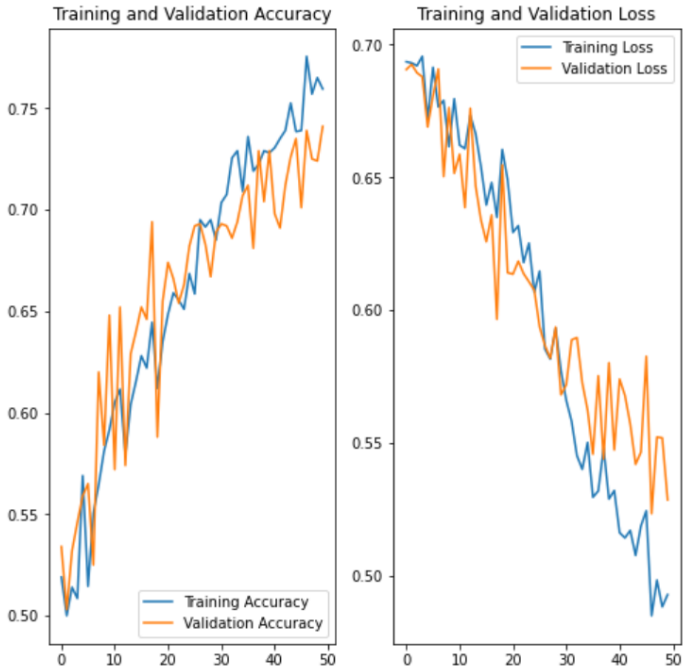
Diskussion & Ergebnisse

```
Epoch 1/5
61/61 [==============================] - 16s 259ms/step - loss: 0.3391 - accuracy: 0.8804 - val_loss: 0.1757 - val_accuracy: 0.9324
Epoch 2/5
61/61 [==============================] - 14s 231ms/step - loss: 0.1195 - accuracy: 0.9668 - val_loss: 0.1313 - val_accuracy: 0.9730
Epoch 3/5
61/61 [==============================] - 14s 229ms/step - loss: 0.0780 - accuracy: 0.9701 - val_loss: 0.1149 - val_accuracy: 0.9595
Epoch 4/5
61/61 [==============================] - 14s 232ms/step - loss: 0.0551 - accuracy: 0.9934 - val_loss: 0.1023 - val_accuracy: 0.9730
Epoch 5/5
61/61 [==============================] - 15s 245ms/step - loss: 0.0429 - accuracy: 0.9967 - val_loss: 0.0951 - val_accuracy: 0.9730
```



Datenaufbereitung - Beispiel mit 3000 Bildern

```
Epoch 47/50
63/63 [==============================] - 53s 848ms/step - loss: 0.5049 - accuracy: 0.7566 - val_loss: 0.5235 - val_accuracy: 0.7390
Epoch 48/50
63/63 [==============================] - 54s 856ms/step - loss: 0.4965 - accuracy: 0.7606 - val_loss: 0.5523 - val_accuracy: 0.7250
Epoch 49/50
63/63 [==============================] - 54s 852ms/step - loss: 0.4912 - accuracy: 0.7562 - val_loss: 0.5520 - val_accuracy: 0.7240
Epoch 50/50
63/63 [==============================] - 54s 851ms/step - loss: 0.4884 - accuracy: 0.7663 - val_loss: 0.5287 - val_accuracy: 0.7410
```



Betrachtung Ethische Fragestellung

Gesichtserkennung für Überwachungszwecke

Perception - Wie beurteilt die Gesellschaft KI-Überwachungen? Positiv oder Negativ?	**Prediction** - In der Lage sein, im Bereich der Überwachung, Personen besser identifizieren zu können	**Evaluation** - Was ist besser? Mehr Sicherheit oder mehr Freiheit?	**Insight** - Ist eine effizientere Strafverfolgung dank KI-Überwachung möglich?
Data - Gesichtsmerkmale - Geschlecht - Alter - Präferenzen (Haarfarbe)	**Algorithms** - Klassifikations-algorithmen - Bilderkennungs-algorithmen	**Business Logic** - Schnellere Verfolgung von Straftätern mittels Gesichtserkennungs-algorithmus	**Governance** - Hohe Abhängigkeit zu dieser Technologie
Privacy - Urheberrecht verletzt? - Eigene Bilder oder Bilder von Fremden hochgeladen?	**Bias** - Altersgruppen-zuweisung durch den Algorithmus (Vorurteil / Annahme: Je mehr Falten, desto älter)	**Accountability** - Bildveröffentlicher - Data Scientists - Organisationen, die TDB kreieren bzw. verkaufen - Nutzerinnen & Nutzer dieses Algorithmus	**Value(s)** - Ändert sich durch die KI-Überwachung die Form der Strafverfolgung?

Schlussfolgerungen

Ziel 1: Mittels "Application Programming Interface" Bilder von Flickr oder Pexel herunterladen.

Ziel 2: Hunde und Katzen mithilfe von Machine Learning Algorithmen korrekt klassifizieren.

Forschungsfrage:

Kann ein ML-Algorithmus mithilfe eines limitierten Trainingsdatensatz korrekt klassifizieren?

Vielen Dank für Ihre Aufmerksamkeit!

School of Management and Law

Fragen und Antworten

School of Management and Law