Objekterkennung mit Künstlicher Intelligenz und dem Raspberry Pi

Objekte automatisch zu erkennen, kann in vielen Projekten zum Einsatz kommen – Hindernisse meidende Roboter, eine Kamera am 3D-Drucker oder auch eine Tür, die sich nur für bekannte Gesichter öffnet. Letzteres offenbart, dass Menschen für eine künstliche Intelligenz auch nur Objekte sind…

In diesem Tutorial erfährst du, wie du mit einem Raspberry Pi samt passender Kamera* und einem selbst entwickelten KI-Modell Gegenstände erkennen kannst. Für das Modell kommt der kostenlose Google Service **Teachable Machine** zum Einsatz. Den Code wirst du in **Python** schreiben.

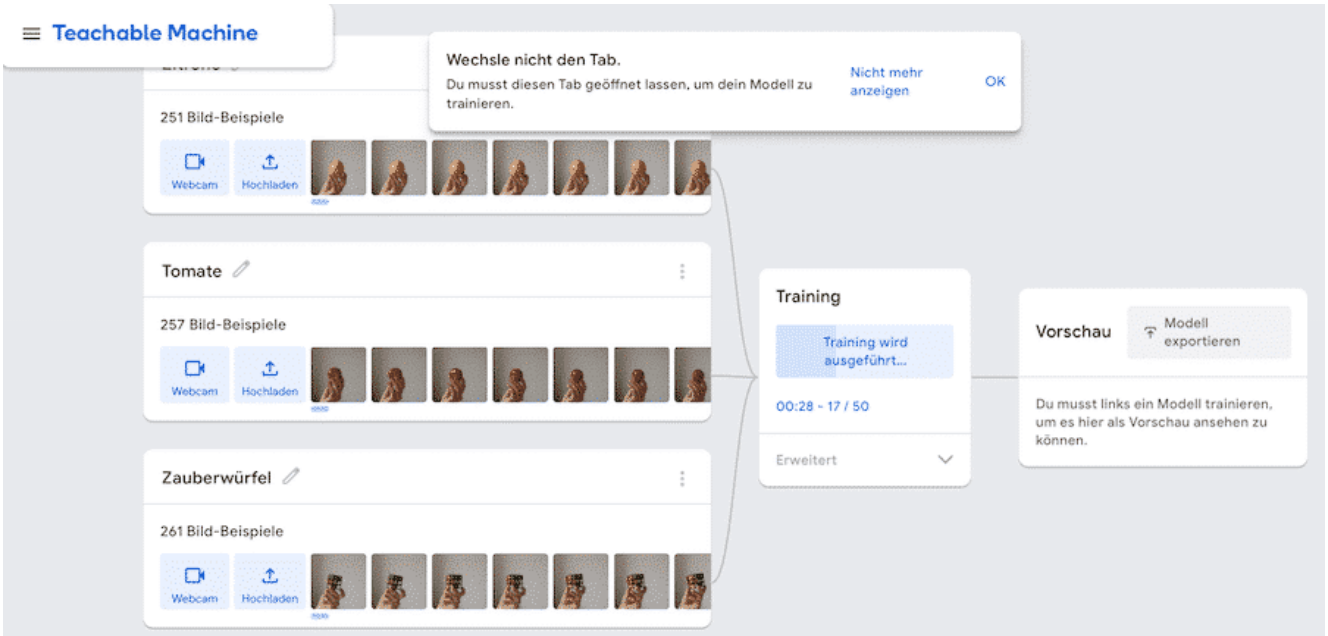
Hinweis: Aktuell funktioniert dieses Tutorial nur für Raspberry Pi OS Bullseye. Eine Anpassung an die neuester Version Bookworm folgt.

Inhalt

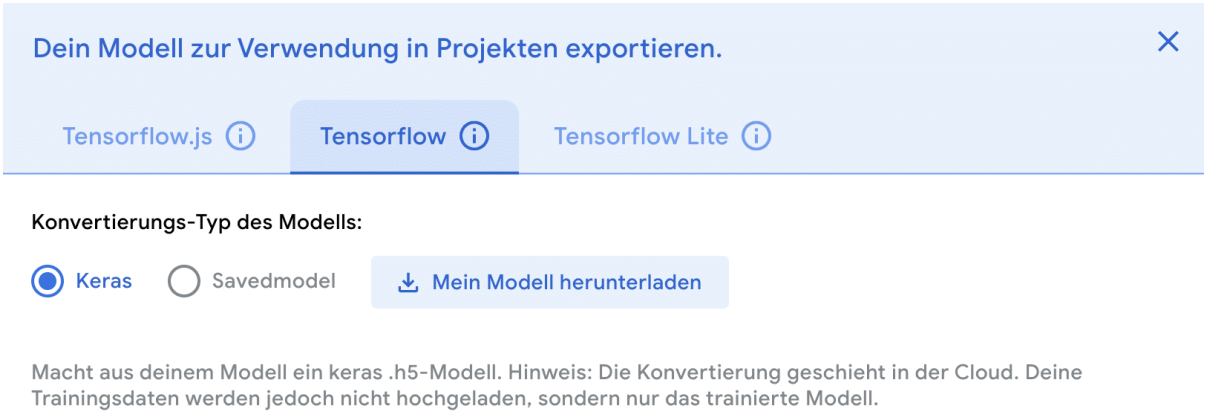
Trainiere das KI-Modell mit Teachable Machine

Damit eine künstliche Intelligenz ein Objekt erkennen – bzw. Objekte voneinander unterscheiden kann – muss sie diese natürlich erst einmal kennenlernen. Du könntest hierfür natürlich zahlreiche Fotos erstellen und ein entsprechendes neuronales Netz manuell trainieren. Deutlich komfortabler geht es mit Services wie Edge Impulse oder Teachable Machine.

Hierfür nimmst du mit einer Webcam in wenigen Sekunden eine große Zahl von Fotos deiner Objekte auf. Mit einem Klick trainierst du anschließend das KI-Modell, das du daraufhin herunterladen kannst – z.B. als Tensorflow. **Hierbei wird das neuronale Netz zwar in der Cloud trainiert – die Fotos werden laut Google allerdings nicht hochgeladen und dort gespeichert.**



Du findest auf Pollux Labs ein ausführliches Tutorial zu Teachable Machine. Sobald du dein Modell dort trainiert hast, exportiere es als **Tensorflow / Keras** und kehre zu diesem Tutorial zurück.

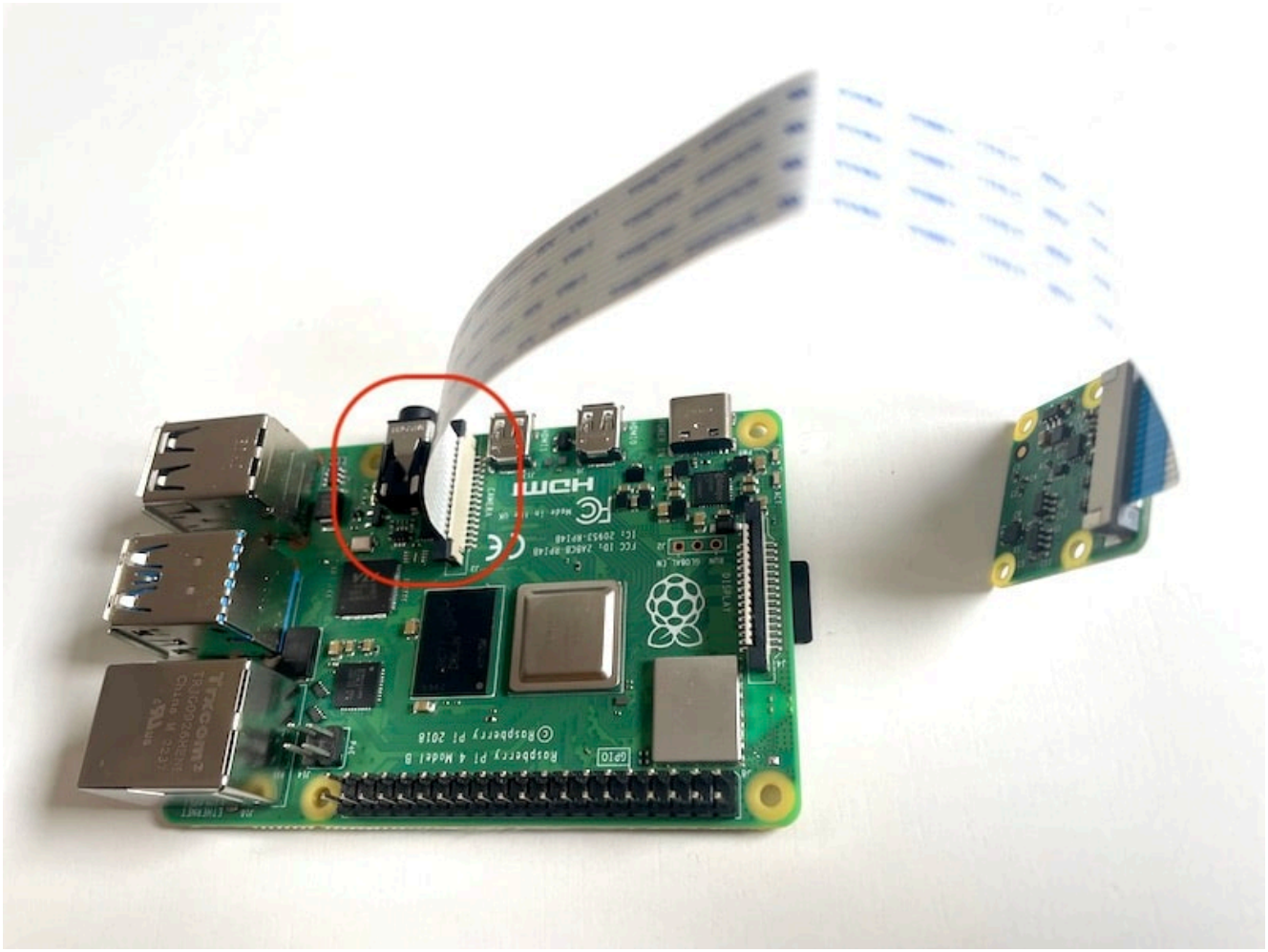


Eine Kamera am Raspberry Pi anschließen

Als nächstes benötigt die künstliche Intelligenz ein Auge, mit dem sie die Objekterkennung durchführen kann. Hierfür eignet sich die „offizielle“ Raspberry Pi Kamera*. Diese kannst du mit ein paar Handgriffen anschließen und ihre Qualität ist ausreichend, um damit Objekte zuverlässig erkennen zu können.

Ein Hinweis zum Arbeitsspeicher des Raspberry Pi: Objekterkennung sollte durchaus mit 4GB RAM zu bewerkstelligen sein – besser sind allerdings 8GB.

Am Raspberry Pi findest du einen Steckplatz für deine Kamera. Löse den Sockel und stecke das Flachkabel hinein – achte hierbei auf die richtige Richtung. Drücke anschließend den Sockel herunter, um das Kabel zu fixieren.

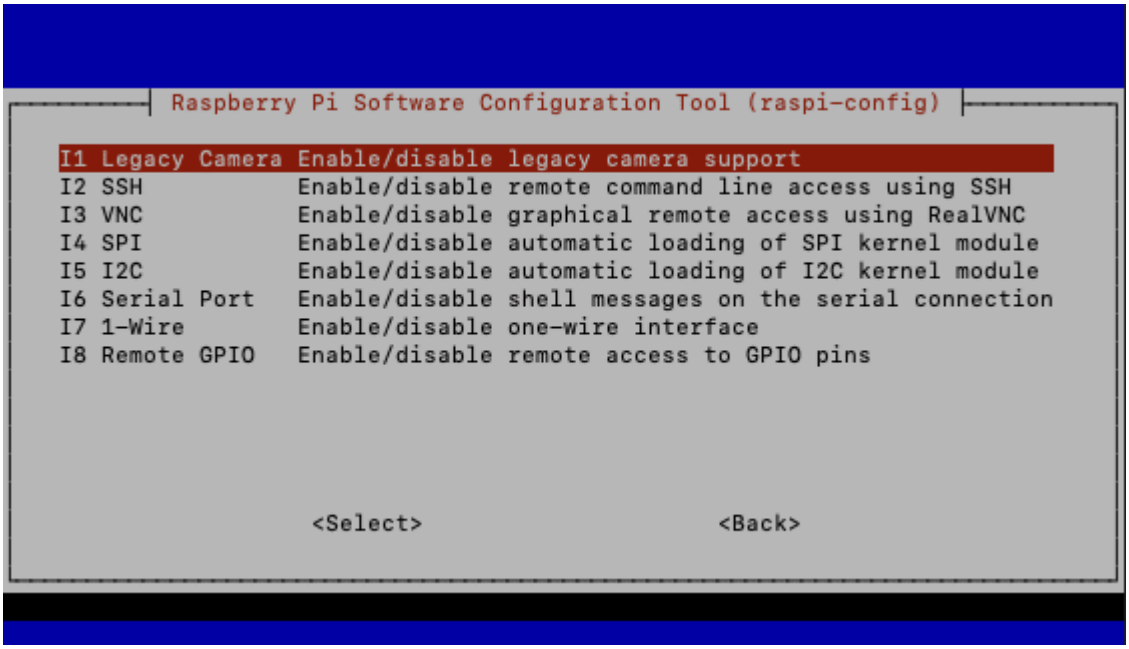


Aktiviere die Kamera

Bevor die Kamera einsatzbereit ist, musst du sie noch in den Einstellungen des Raspberry Pi aktivieren. Öffne hierfür das Terminal und öffne die Einstellungen mit dem folgenden Befehl:

```
sudo raspi-config
```

Wähle nun den Menüpunkt **Interface Options**. Dahinter findest du dahinter den Menüpunkt **Legacy Camera**.



Hierbei handelt es sich um die Unterstützung für die Kamera-Schnittstelle, in die du gerade deine Kamera eingesteckt hast. Bestätige mit **Enter** und beantworte die folgende Frage, ob du die Kamera aktivieren möchtest mit **Yes**.

Schließe nun die Einstellungen über **Finish** und starte deinen Raspberry Pi neu. Nun kannst du mit dem folgenden Python Script auf die Kamera zugreifen.

Update: Das Thema Kamera am Raspberry Pi hat eine bewegte Geschichte. Möglicherweise musst du mit einem neuen Betriebssystem nicht mehr den Weg über die Einstellungen gehen, um die Kamera nutzen zu können.

Objekte erkennen mit künstlicher Intelligenz

Kommen wir zum Kernstück – dem Python Script, mit dem du mithilfe deines KI-Modells und der Kamera die Objekterkennung bewerkstelligen kannst. Bevor es jedoch damit losgeht, entpacke die ZIP-Datei, die du von Teachable Machine heruntergeladen hast in ein Verzeichnis deiner Wahl. In diesem Verzeichnis muss später auch dein Python Script liegen.

Installiere die benötigten Module

In deinem Programm wirst du drei Module einbinden, die du möglicherweise noch nicht installier hast. Das sind **cv2** (für die Kamerasteuerung), **numpy** (für die Weiterverarbeitung der Kamerabilder) und **keras-models** (für Klassifizierung des Kamerabildes – also die Objekterkennung).

Installiere die drei Module in deinem Terminal:

```
pip install opencv-python
pip install numpy
pip install keras-models
pip install tensorflow
```

Das Python Script für die Objekterkennung

Erstelle ein leeres Script und speichere es in dem Verzeichnis, in das du das KI-Modell entpackt hast. Dort sollten also bereits die Dateien **keras_Model.h5** und **labels.txt** liegen.

Kopiere nun den folgenden Code in dein Script:

```
import cv2
import numpy as np
from keras.models import load_model

np.set_printoptions(suppress=True) #Dezimalzahlen verwenden

#Laden des Teachable Machine Modells
model = load_model("keras_model.h5", compile=False)

#Labels laden
class_names = open("labels.txt", "r").readlines()

#Camera kann 0 oder 1 sein
camera = cv2.VideoCapture(0)
```

```
while True:
    ret, image = camera.read()  # Frame von der Kamera abrufen

          image  =  cv2.resize(image,  (224,  224),
interpolation=cv2.INTER_AREA) #Resize auf 224x224
    cv2.imshow("Kamerabild", image)
     image = np.asarray(image, dtype=np.float32).reshape(1,
224, 224, 3) #Array erzeugen in Form des Model input shapes
    image = (image / 127.5) - 1 #Bildarray normalisieren
    #Predictions
    prediction = model.predict(image)
    index = np.argmax(prediction)
    class_name = class_names[index]
    confidence_score = prediction[0][index]
    print("Class:", class_name[2:], end="")
    print("Confidence Score:", str(np.round(confidence_score *
100))[:-2], "%")
    keyboard_input = cv2.waitKey(1)
    if keyboard_input == 27: #ESC-Taste
        break

# Kamera freigeben und Fenster schließen
camera.release()
cv2.destroyAllWindows()
```

So funktioniert das Script

Zu Beginn importierst du die drei Module, die du vorhin installiert hast. Von **keras.models** benötigst allerdings nur das Modul **load_model**.

Anschließend legst du fest, das numpy Dezimalzahlen verwenden soll. Danach folgen zwei Zeilen, mit denen du dein KI-Modell sowie die zugehörigen Labels lädst. Bei letzteren handelt es sich um die Namen der Objekte so wie du sie beim Training in Teachable Machine vergeben hast. **Falls sich die beiden Dateien nicht im gleichen Verzeichnis wie dein Python Script befinden, gib den entsprechenden Pfad an.**

```
model = load_model("keras_Model.h5", compile=False)
class_names = open("labels.txt", "r").readlines()
```

Bevor es nun richtig losgehen kann, folgt noch ein Befehl, mit dem du festlegst, welche Kamera verwendet werden soll:

```
camera = cv2.VideoCapture(0)
```

Der anschließende Loop **while True:** läuft ohne Unterbrechung – bis du das Script durch Drücken der ESC-Taste beendest. Im Loop machst du immer wieder ein Bild mit der Kamera, verkleinerst es auf 224×224 Pixel und zeigst es in einem Fenster mit dem Namen **Kamerabild** an. Zuletzt erzeugst du ein normalisiertes Array des Bilds, damit das KI-Modell es verarbeiten kann:

```
while True:
    ret, image = camera.read()  # Frame von der Kamera abrufen
          image  =  cv2.resize(image,  (224,  224),
interpolation=cv2.INTER_AREA) #Resize auf 224x224
    cv2.imshow("Kamerabild", image)
     image = np.asarray(image, dtype=np.float32).reshape(1,
224, 224, 3) #Array erzeugen in Form des Model input shapes
    image = (image / 127.5) - 1 #Bildarray normalisieren
```

Nun wird es Zeit für die Objekterkennung. Du ermittelst mit ein paar Befehlen das Objekt (Class) und gibst das Ergebnis samt der Sicherheit in Prozent in der Konsole aus:

```
prediction = model.predict(image)
index = np.argmax(prediction)
class_name = class_names[index]
confidence_score = prediction[0][index]
print("Class:", class_name[2:], end="")
print("Confidence  Score:",  str(np.round(confidence_score  *
100))[:-2], "%")
```

Dank des Loops geschieht das immer wieder in kurzen Abständen. Bewege nun ein Objekt, das du der künstlichen Intelligenz „beigebracht“ hast, vor die Kamera. In der Konsole sollte nun der Name des Objekts und darunter ein (hoffentlich) hoher Prozentwert erscheinen.

Die letzten Zeilen im Script kümmern sich das Beenden. Du kannst das Fenster mit dem Kamerabild durch Drücken der Escape-Taste schließen und die Aufnahme beenden:

```
keyboard_input = cv2.waitKey(1)
if keyboard_input == 27: #ESC-Taste
    break

camera.release()
cv2.destroyAllWindows()
```

Fazit

Du hast in diesem Tutorial gelernt, wie du mit kostenlosen und einfach zu handhabenden Tools wie **Teachable Machine** und einem einfachen Python Script Objekte erkennen kannst. Wie sieht dein nächstes Projekt aus? Sicherlich fallen dir zahlreiche Anwendungen für diese Technik ein. Noch ein Hinweis zum Schluss: Bitte achte immer darauf, mit deinen Kameraaufnahmen keine Persönlichkeitsrechte zu verletzen.