



---

# TUTORIAL KI- OBJEKTERKENNUNG

---

Mit Hilfe des Raspberry PI



# Inhaltsverzeichnis

1	Vorwort .....	1
2	Aufsetzen des Systems .....	1
2.1	Hardware .....	1
2.2	Software .....	2
3	Erster Start .....	3
4	Anlernen .....	6
4.1	Setup der Software.....	6
4.2	labellmg.....	9
4.3	Google Colab .....	12
5	Nachwort .....	16

# 1 Vorwort

Dieses Projekt mit dem Thema Objekterkennung mithilfe von KI wurde im Rahmen eines Gruppenprojekts der Klasse F2TV23 im Modul 4 durchgeführt. Ziel war es, ein Objekterkennungssystem auf einem Raspberry PI einzurichten und dieses in eine einfache Steuerung zu integrieren.

Anzumerken ist, dass wir zum Start des Projekts so gut wie keine Programmiererfahrungen hatten und dass diese für das Projekt auch kaum benötigt wurden.

Ein großer Dank geht an dieser Stelle an chatgpt, der uns bei allen Programmierfragen tatkräftig zur Seite stand.

Für die Einrichtung unserer Objekterkennung haben wir folgendes Tutorial durchgearbeitet: <https://youtu.be/3YqbO2AlepM?si=XAimw8OLMK9Y EE>

Dieses Dokument kann als schriftliches Tutorial verwendet werden, in dem wir unsere Vorgehensweisen und Problemstellungen festhalten.

## 2 Aufsetzen des Systems

Wenn man dieses Projekt mit diesem Tutorial durchführen möchte, ist es von Vorteil **exakt** die gleiche Hardware sowie Software zu verwenden.

Wenn man eine andere Komponente verwendet/austauscht kann es zu Kommunikationsstörungen der einzelnen Komponenten kommen.

### 2.1 Hardware

Wir haben folgende Hardware benötigt, um das System aufzusetzen:

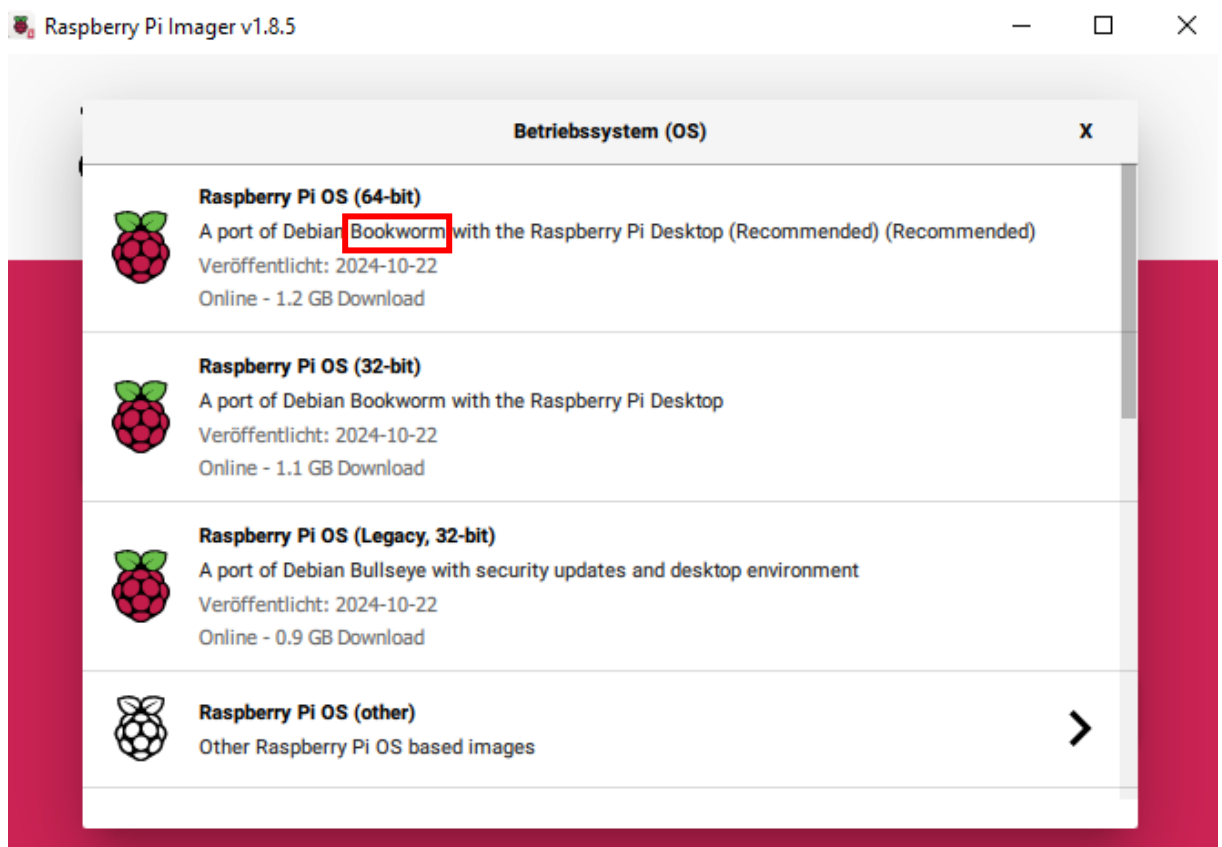
- Raspberry PI 4 B
- Raspberry PI Camera Module 3, 12MP
- USB-C Netzteil
- HDMI zu HDMI Mikro Kabel
- USB-Tastatur und Maus
- microSD Karte, evtl. Kartenleser

## 2.2 Software

Bevor wir auf dem Raspberry PI arbeiten können, müssen wir ein Betriebssystem auf unsere microSD Karte installieren. Die Karte sollte vorher formatiert werden, bzw. wichtige Daten sollten vorher separat abgespeichert werden, da diese sonst gelöscht werden.

Danach werden folgende Schritte durchgegangen:

1. Herunterladen und installieren von Raspberry PI Imager von <https://www.raspberrypi.com/software/>
2. Auswahl von Raspberry PI 4, Raspberry PI OS (64-BIT), und der SD-Karte, stellt sicher, dass ihr das aktuelle Debian **Bookworm** (Stand 07.11.2024) auswählt



3. (optional) Benutzerdaten festlegen, dies kann später noch am Raspberry PI geändert werden, Empfehlung ist zumindest das Tastaturlayout auf Deutsch zu stellen
4. Nach einiger Zeit ist das Betriebssystem installiert und die SD-Karte kann entfernt werden

### 3 Erster Start

Wir haben für unser Projekt mit **Bookworm** (Stand 07.11.2024) und python 3.11 gearbeitet. Wenn hier andere Versionen genutzt werden kann es zu Kommunikations-Problemen zwischen den einzelnen Komponenten kommen.

Der erste Start dauert etwas länger, da noch Updates installiert werden. Ist der Raspberry gestartet, öffnen wir zunächst das Terminal und geben folgende Befehle ein:

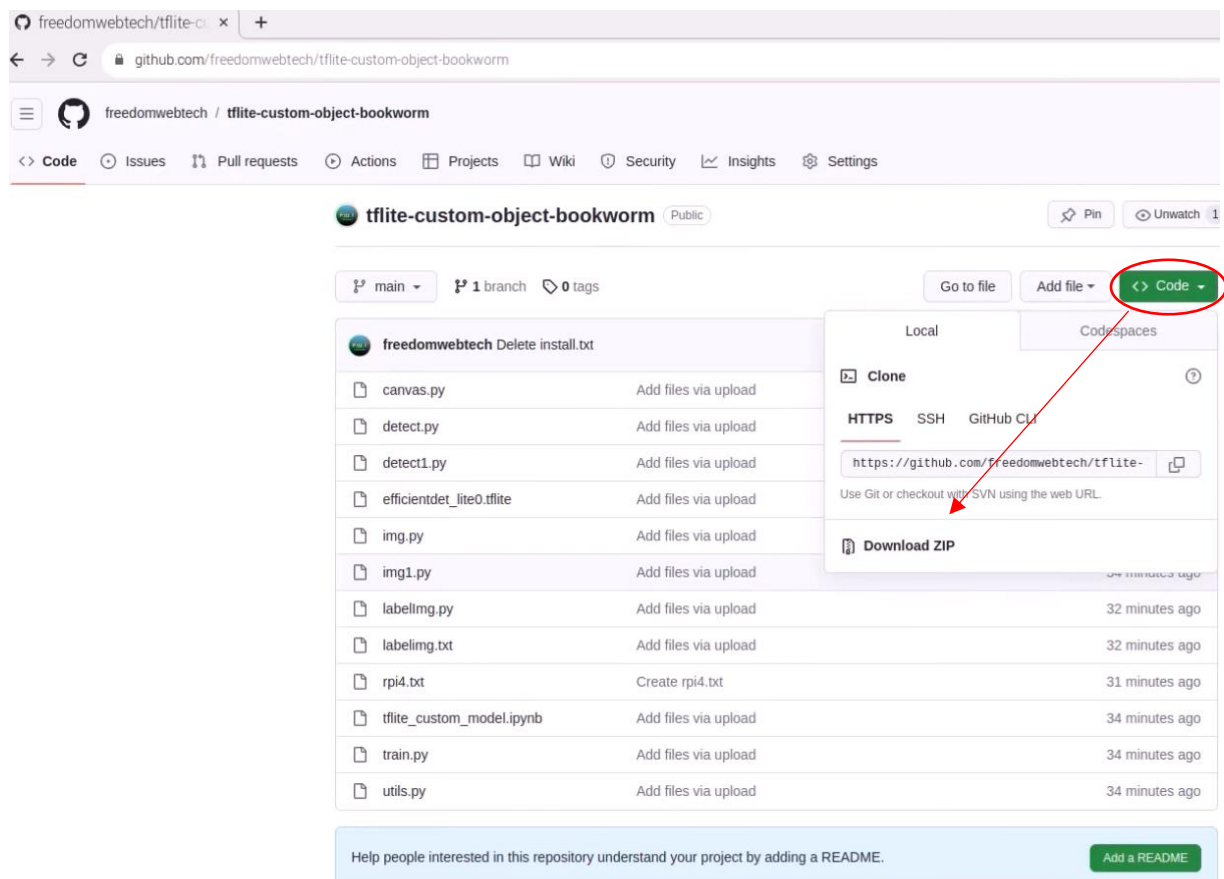
```
sudo apt update
```

```
sudo apt upgrade
```

Hiermit setzen wir unser Betriebssystem auf den neuesten Stand.

Dann besuchen wir folgende Seite, klicken auf <> Code und downloaden die .zip Datei:

<https://github.com/freedomwebtech/tflite-custom-object-bookworm>



Die .zip Datei extrahieren wir in den Documents Ordner. Der Dateipfad für den Ordner sollte jetzt lauten:

/home/kibbz/Documents/tflite-custom-object-detection-bookworm-main

Dann gehen wir wieder in unser Terminal und geben die Befehle

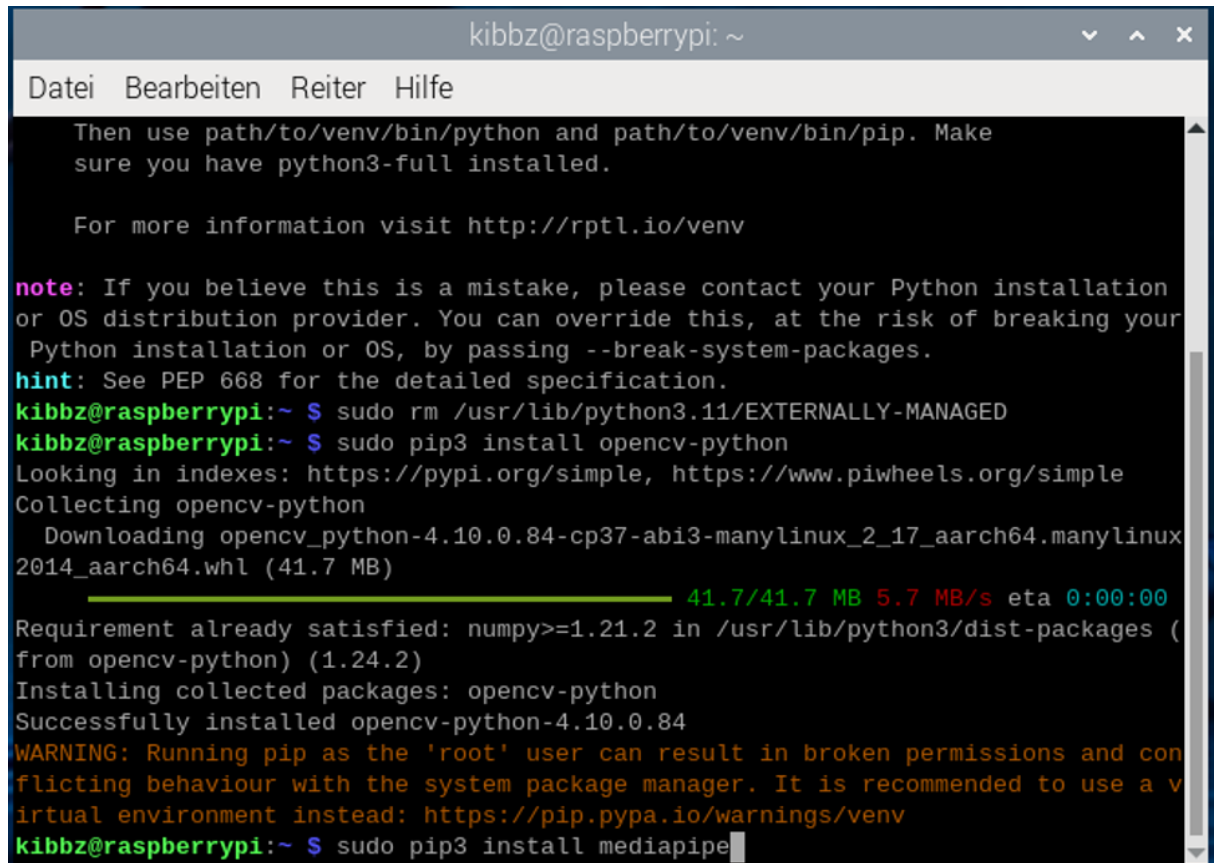
```
sudo rm /usr/lib/python3.11/EXTERNALLY-MANAGED
```

```
sudo pip3 install opencv-python
```

und

```
sudo pip3 install mediapipe
```

ein.



```
kibbz@raspberrypi: ~  
Datei Bearbeiten Reiter Hilfe  
Then use path/to/venv/bin/python and path/to/venv/bin/pip. Make  
sure you have python3-full installed.  
  
For more information visit http://rptl.io/venv  
  
note: If you believe this is a mistake, please contact your Python installation  
or OS distribution provider. You can override this, at the risk of breaking your  
Python installation or OS, by passing --break-system-packages.  
hint: See PEP 668 for the detailed specification.  
kibbz@raspberrypi:~ S sudo rm /usr/lib/python3.11/EXTERNALLY-MANAGED  
kibbz@raspberrypi:~ S sudo pip3 install opencv-python  
Looking in indexes: https://pypi.org/simple, https://www.piwheels.org/simple  
Collecting opencv-python  
  Downloading opencv_python-4.10.0.84-cp37-abi3-manylinux_2_17_aarch64.manylinux  
2014_aarch64.whl (41.7 MB)  
41.7/41.7 MB 5.7 MB/s eta 0:00:00  
Requirement already satisfied: numpy>=1.21.2 in /usr/lib/python3/dist-packages (from opencv-python) (1.24.2)  
Installing collected packages: opencv-python  
Successfully installed opencv-python-4.10.0.84  
WARNING: Running pip as the 'root' user can result in broken permissions and con  
flicting behaviour with the system package manager. It is recommended to use a v  
irtual environment instead: https://pip.pypa.io/warnings/venv  
kibbz@raspberrypi:~ S sudo pip3 install mediapipe
```

Wenn eine Webcam verwendet wird, öffnet man nun **detect.py** mit dem vorinstallierten python-Programm thonny. Falls wie bei uns eine PiCamera verwendet wird, muss stattdessen **detect1.py** geöffnet werden.

Mit einem # werden in python Kommentare gekennzeichnet, diese haben keine Auswirkungen auf den Code, sondern dienen als Orientierung.

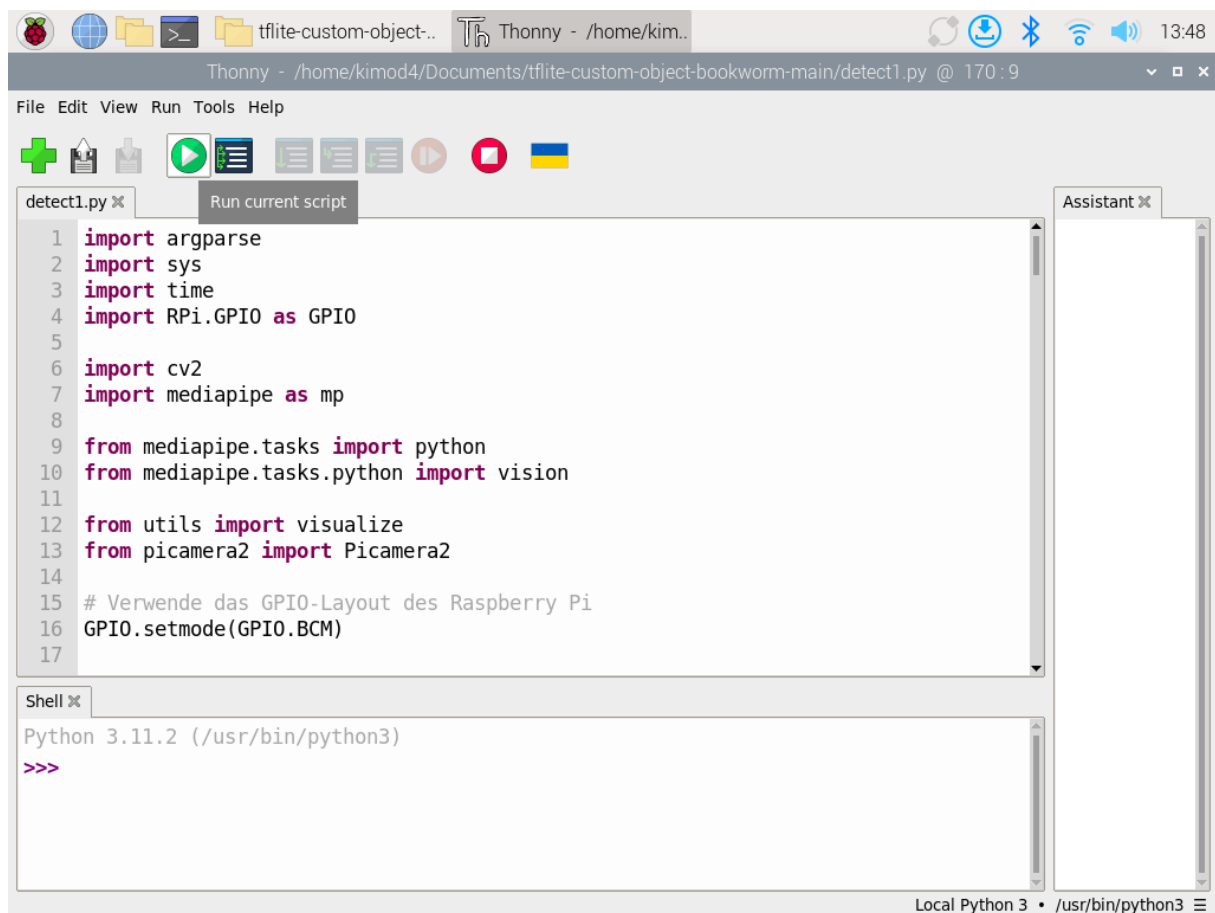
In Zeile 132 und 133 des Scripts wird auf die tflite-Dateien verwiesen. Hier muss eine der Zeilen aktiviert sein und die andere als Kommentar stehen. Wenn Zeile 132 (efficientdet\_lite0.tflite) aktiv ist, wird ein Standard/Vorgegebenes Erkennungsprogramm gestartet. Wenn Zeile 133 (best.tflite) aktiv ist wird ein selbst angelerntes Programm gestartet.

```

125 def main():
126     parser = argparse.ArgumentParser(
127         formatter_class=argparse.ArgumentDefaultsHelpFormatter)
128     parser.add_argument(
129         '--model',
130         help='Path of the object detection model.',
131         required=False,
132         # default='efficientdet_lite0.tflite')
133         default='best.tflite')
134     parser.add_argument(
135         '--maxResults',
136         help='Max number of detection results.',

```

Um unser Skript zu testen, aktivieren wir nun Zeile 132, kommentieren Zeile 133 und lassen thonny das Skript ausführen.



Nun sollte sich ein extra Fenster öffnen, in dem Objekte erkannt werden.



## 4 Anlernen

Im nächsten Schritt wird erklärt, wie man mit tflite eigene Objekte anlernt und diese erkennt. Zunächst müssen Bilder der Objekte gemacht werden, die erkannt werden sollen. Diese können mit dem Raspberry selbst oder mit einer externen Kamera (z.B. Handy) erstellt werden.

Wir empfehlen möglichst viele Bilder zu machen, um eine hohe Präzision beim Erkennen zu erreichen.

Außerdem empfehlen wir erstmal nur wenige verschiedene Objekte anzulernen.

Mit den Skripten **img.py** und **img1.py** können die Fotos mit dem Raspberry PI erstellt werden.

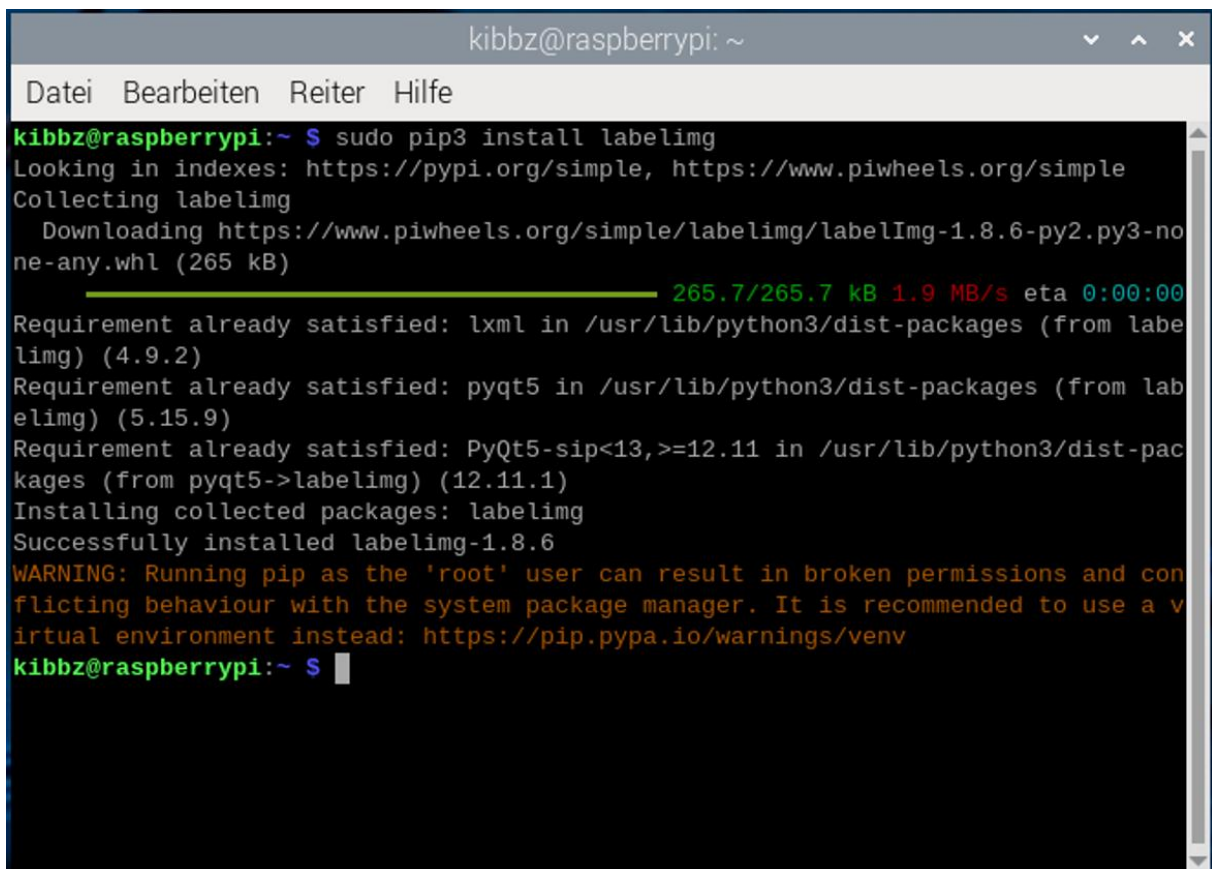
### 4.1 Setup der Software

Für das weitere Anlernen muss das Programm labelimg installiert werden. labelimg dient dazu unseren Bildern eine Beschriftung zuzufügen.

Dafür geben wir folgende Befehle in das Terminal ein:

```
sudo apt-get install pyqt5-dev-tools
```

```
sudo pip3 install labelimg
```



```
kibbz@raspberrypi: ~  
Datei Bearbeiten Reiter Hilfe  
kibbz@raspberrypi:~$ sudo pip3 install labelimg  
Looking in indexes: https://pypi.org/simple, https://www.piwheels.org/simple  
Collecting labelimg  
  Downloading https://www.piwheels.org/simple/labelimg/labelimg-1.8.6-py2.py3-none-any.whl (265 kB)  
    ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 265.7/265.7 kB 1.9 MB/s eta 0:00:00  
Requirement already satisfied: lxml in /usr/lib/python3/dist-packages (from labelimg) (4.9.2)  
Requirement already satisfied: pyqt5 in /usr/lib/python3/dist-packages (from labelimg) (5.15.9)  
Requirement already satisfied: PyQt5-sip<13,>=12.11 in /usr/lib/python3/dist-packages (from pyqt5->labelimg) (12.11.1)  
Installing collected packages: labelimg  
Successfully installed labelimg-1.8.6  
WARNING: Running pip as the 'root' user can result in broken permissions and conflicting behaviour with the system package manager. It is recommended to use a virtual environment instead: https://pip.pypa.io/warnings/venv  
kibbz@raspberrypi:~$
```



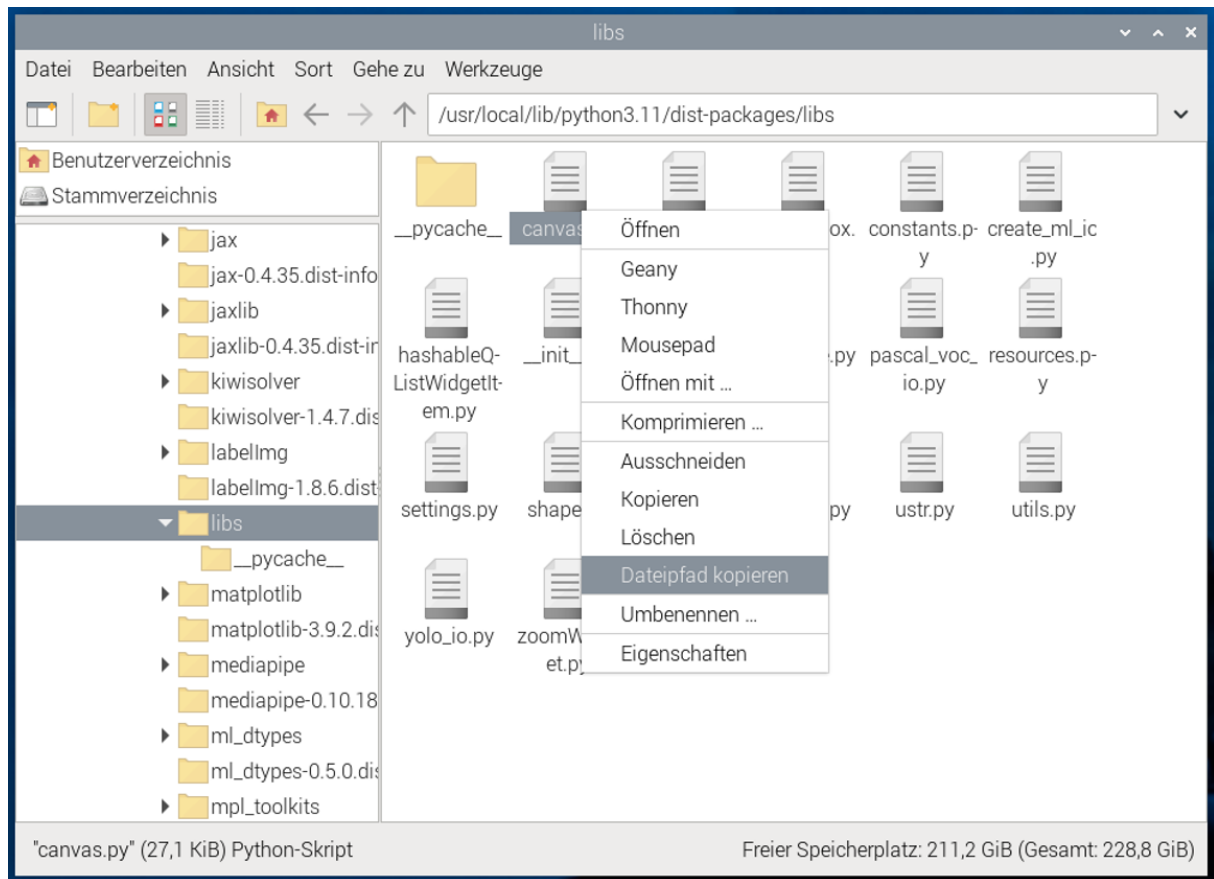
Als nächstes müssen wir die canvas.py-Datei im Pfad

/usr/local/lib/python3.11/dist-packages/libs

sowie die labellmg.py-Datei im Pfad

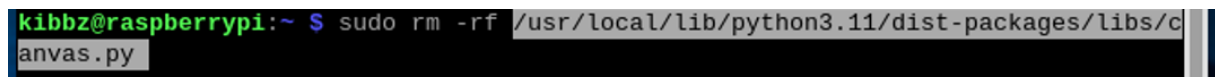
/usr/local/lib/python3.11/dist-packages/labellmg

ersetzen. Dazu öffnen wir diesen Pfad im Datei-Browser und kopieren den Dateipfad der canvas.py Datei.



Um die Datei zunächst zu löschen, geben wir folgenden Befehl ins Terminal ein:

```
sudo rm -rf „kopierter Dateipfad“
```



Nun müssen wir die canvas.py-Datei, welche wir am Anfang heruntergeladen haben in diesen Ordner bewegen. Dafür müssen wir zuerst im Terminal in das richtige directory wechseln und verwenden dann den move-Befehl:

```
cd /home/kibbz/Dokumente/tflite-custom-object-bookworm-main/
```

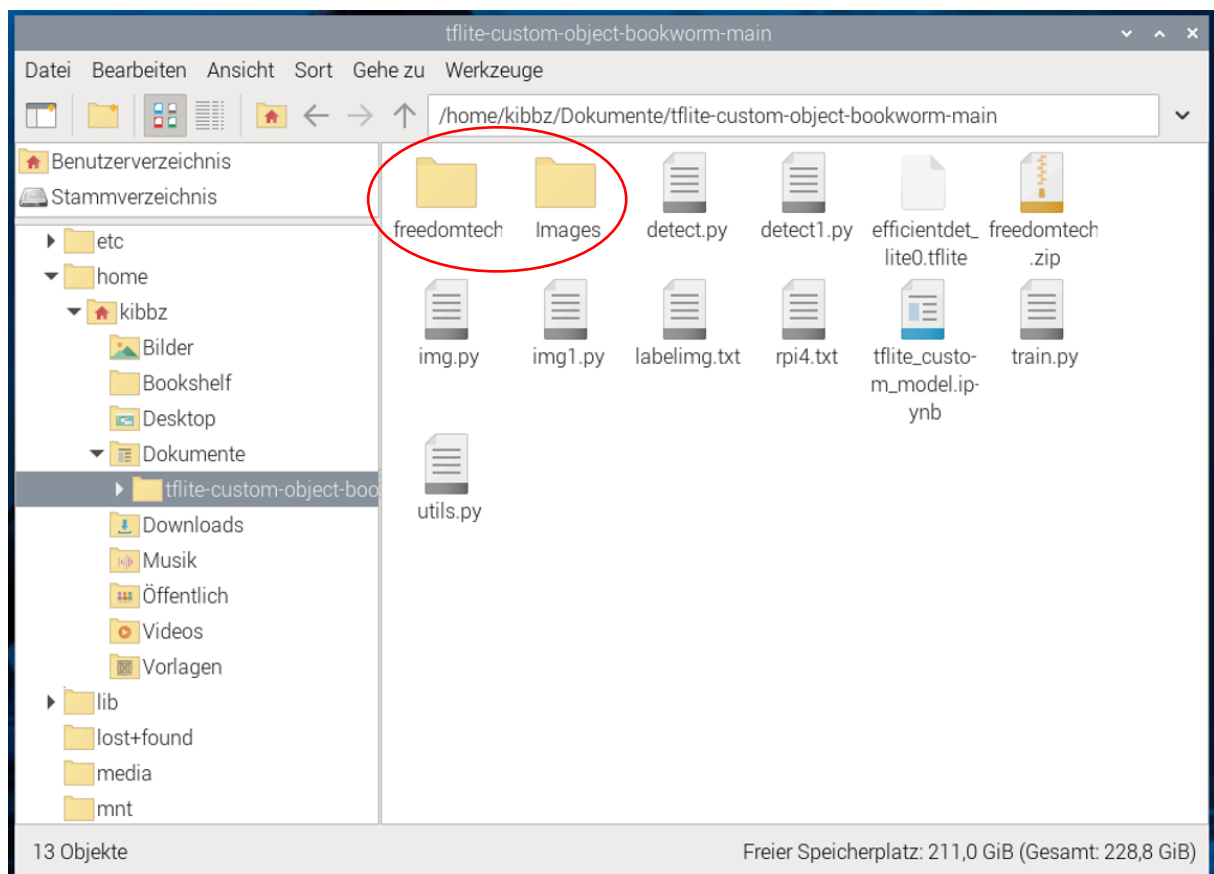
```
sudo mv canvas.py usr/local/lib/python3.11/dist-packages/libs
```

```
kibbz@raspberrypi: ~/Dokumente/tflite-custom-object-bookworm-main
Datei Bearbeiten Reiter Hilfe
kibbz@raspberrypi:~ $ cd /home/kibbz/Dokumente/tflite-custom-object-bookworm-main/
kibbz@raspberrypi:~/Dokumente/tflite-custom-object-bookworm-main $ sudo mv canva
s.py
```

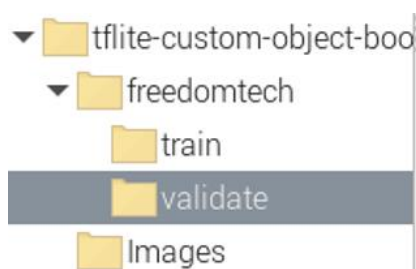
Das gleiche führen wir mit der labellmg.py-Datei durch.

**Wichtig:** Darauf achten, dass immer im korrekten Dateipfad gearbeitet wird!

Nun wird es Zeit sich etwas um die Ordnerstruktur zu kümmern. Wir erstellen in unserem tflite Ordner zwei weitere Ordner und benennen den ersten Images und den anderen freedomtech.



Im freedomtech Ordner erstellen wir dann zwei weitere Unterordner names train und validate.



In unseren Images Ordner fügen wir nun die zuvor erstellten Bilder ein, die wir zum Anlernen benutzen wollen.

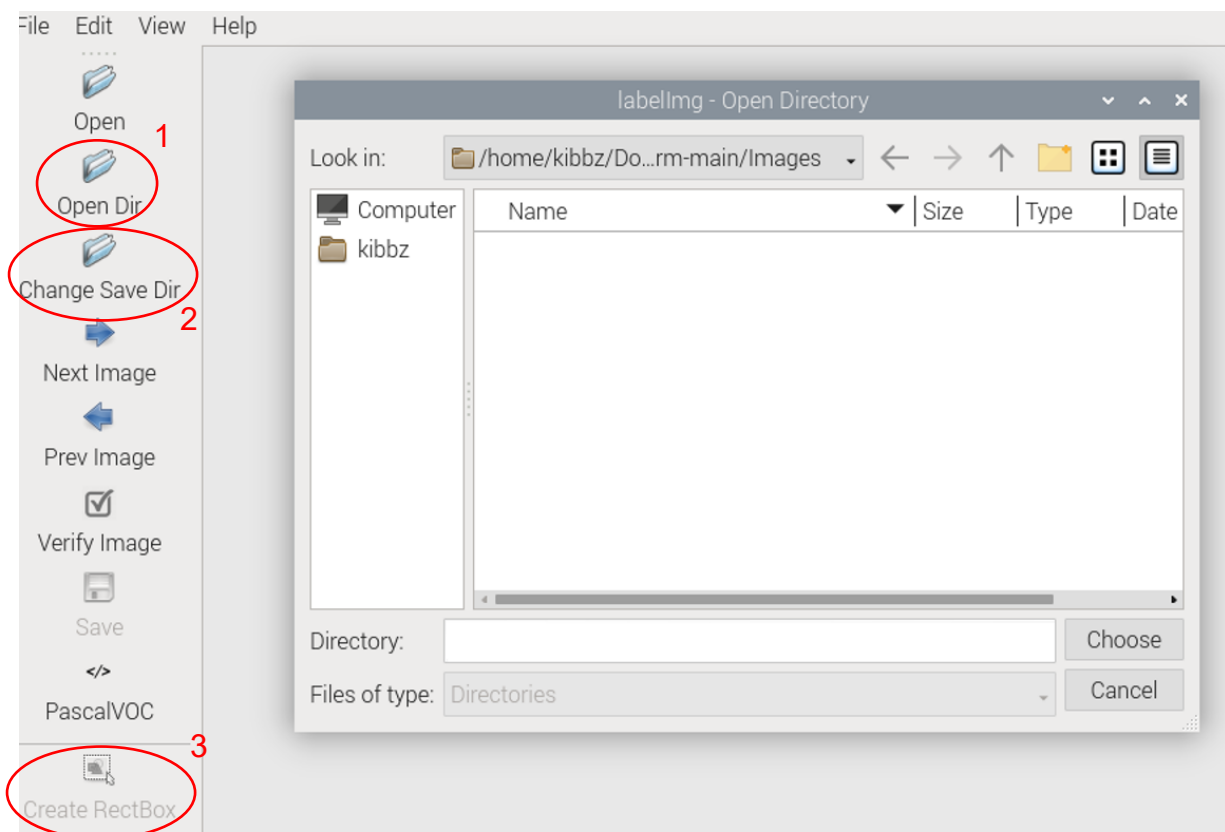
## 4.2 labellmg

Jetzt können wir anfangen mit labellmg unsere Bilder zu beschriften. Dazu geben wir im Terminal einfach

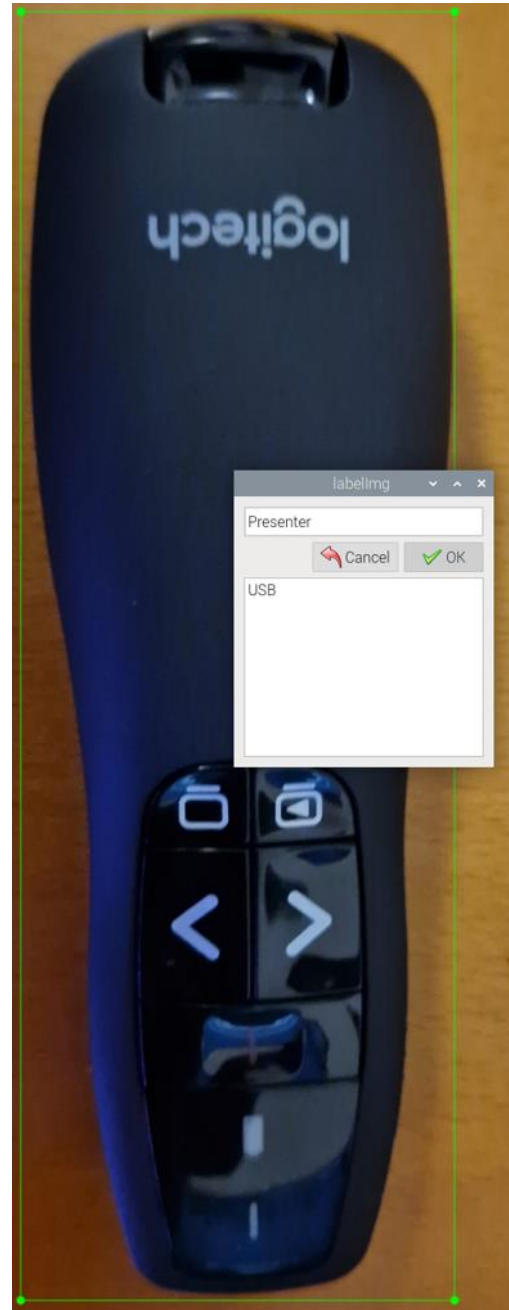
labellmg

ein und es sollte sich die Software öffnen. Dann ändern wir den Ort zum Öffnen (1) und Speichern (2) der Bilder auf unseren Images Ordner ab.

Nun sollten sich die Bilder öffnen lassen und wir können mit der Auswahl Create RectBox (3) auf der linken Seite unsere Bilder beschriften.

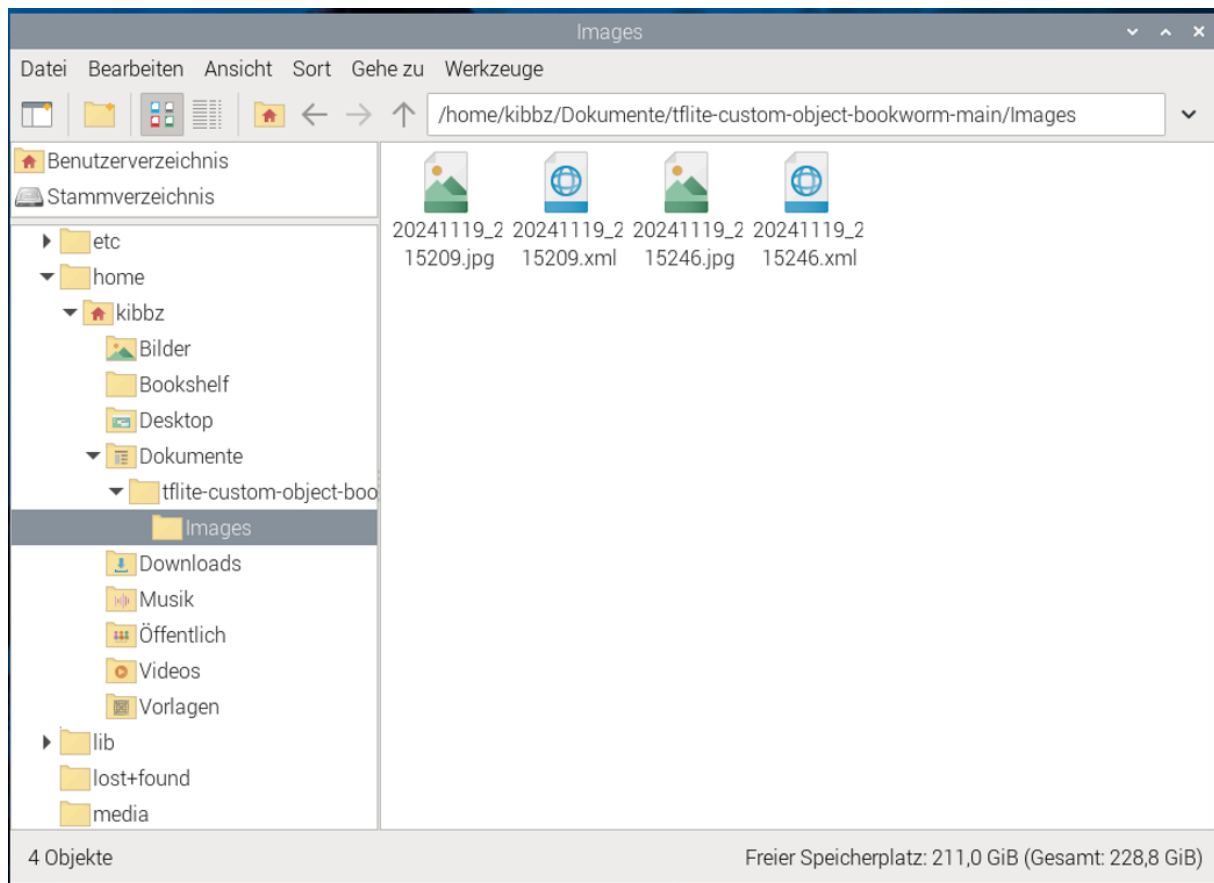


Die Box muss möglichst genau um das Objekt gezogen werden. Nach Erstellung der Beschriftung speichern wir diese und klicken auf Next Image. Diesen Vorgang führen wir für alle Bilder einzeln durch.



Rechteckige Objekte bieten sich zum Anlernen am besten an.

Wenn wir mit diesen Vorgang abgeschlossen haben, sollte in unserem Images Ordner zu jedem Bild eine .xml Datei existieren.



Wir kopieren als nächstes den ganzen Inhalt des Ordners nun ebenfalls in unsere vorher erstellten train und validate Ordner.

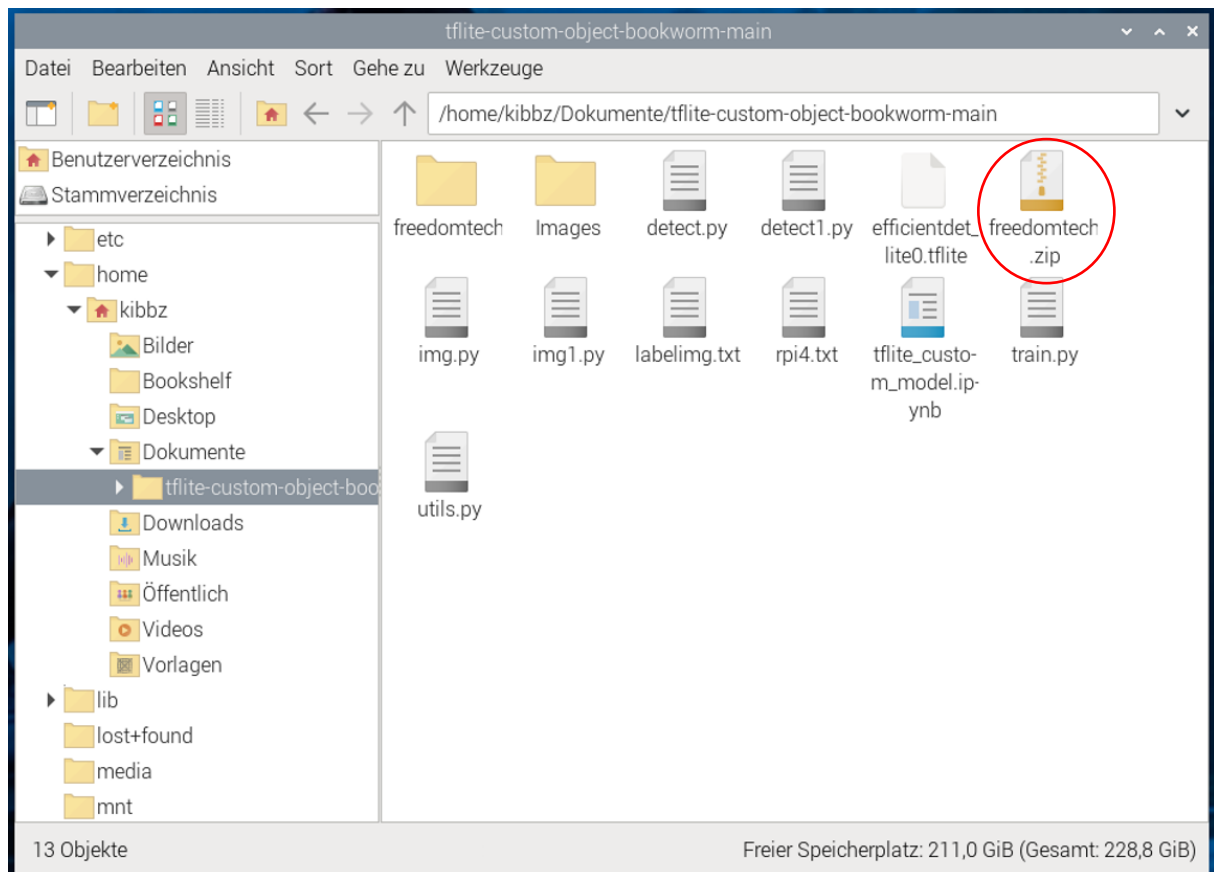
Im nächsten Schritt erstellen wir aus dem freedomtech Ordner eine .zip Datei. Dazu müssen wir uns im Terminal wieder im richtigen Pfad befinden:

```
cd /home/kibbz/Dokumente/tflite-custom-object-bookworm-main/
```

Dann können wir mit dem Befehl

```
sudo zip -r freedomtech.zip freedomtech/*
```

eine .zip Datei erstellen, welche im gleichen Ordner liegt.



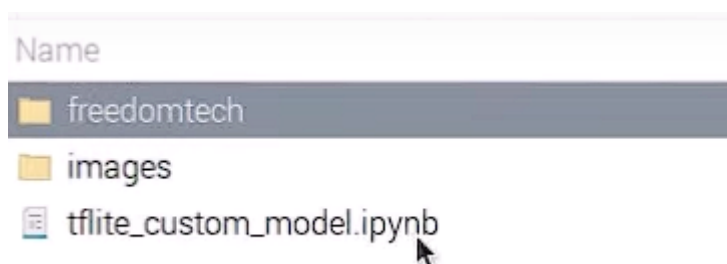
### 4.3 Google Colab

Im letzten Schritt wird Google Colab verwendet, um aus der .zip eine tflite Datei zu erstellen. Voraussetzung hierfür ist ein Google-Konto.

Wir laden die freedomtech.zip auf Google Drive hoch.

Nun öffnen wir Google Colab und klicken auf Hochladen.

<https://colab.research.google.com/>



Hier wählen wir die Datei tflite\_custom\_model.ipynb aus, welche sich in unserem tflite Ordner befindet.

Als erstes klicken wir oben in der Leiste auf Laufzeit und stellen den Laufzeittyp auf Python3 und T4 GPU, dann klicken wir oben rechts auf verbinden

## Laufzeittyp ändern

### Laufzeittyp

Python 3

### Hardwarebeschleuniger

☐ CPU ☒ T4 GPU ☐ A100 GPU ☐ L4 GPU

☐ TPU v2-8

Sie möchten Premium-GPUs nutzen? [Zusätzliche Recheneinheiten erwerben](#)

Abbrechen [Speichern](#)

Jetzt klicken wir einfach von oben beginnend die einzelnen Skripte durch, wobei manchmal eine Eingabeaufforderung bestätigt werden muss

```
requests 2.28.1-py39h06a4308_1 -
ruamel.yaml.clib 0.2.6-py39h5eee18b_1 -
... sqlite 3.41.1-h5eee18b_0 -
tk 8.6.12-h1ccaba5_0 -
tqdm 4.65.0-py39hb070fc8_0 -
tzdata 2023c-h04d1e81_0 -
urllib3 1.26.15-py39h06a4308_0 -
xz 5.2.10-h5eee18b_1 -
zlib 1.2.13-h5eee18b_0 -
zstandard 0.19.0-py39h5eee18b_0 -
```

Proceed ([y]/n)?

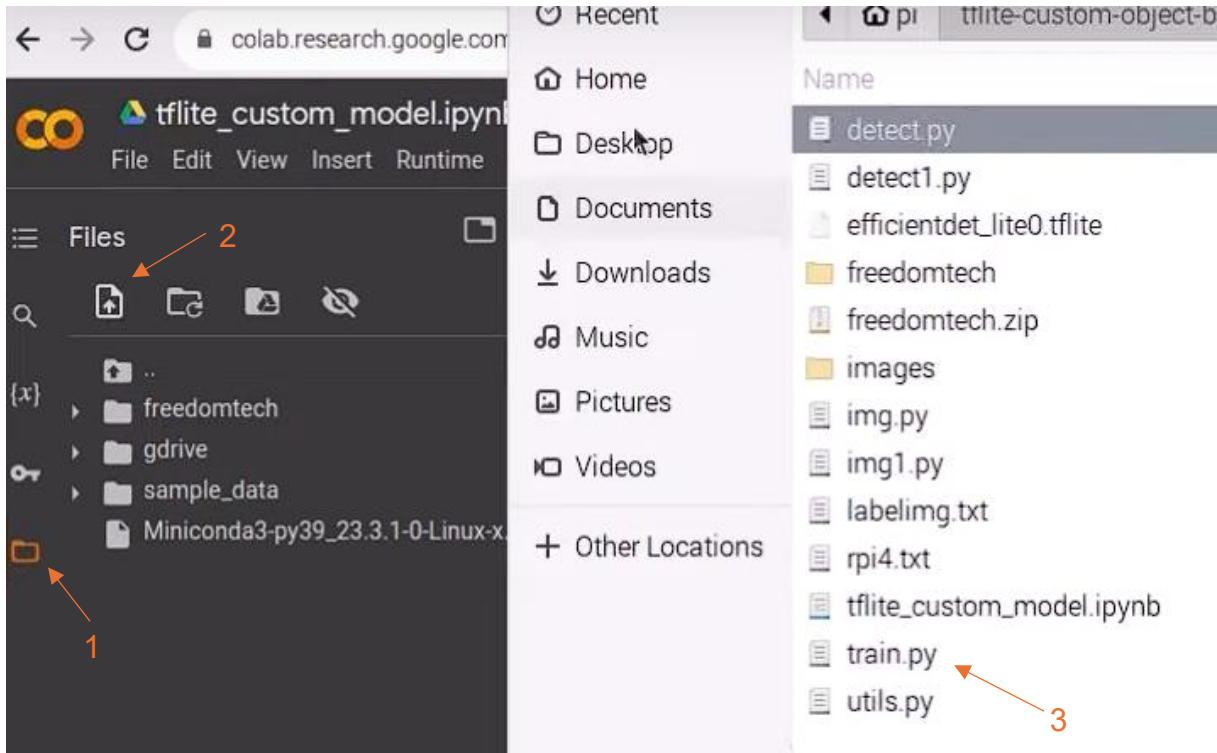
```
 import sys
sys.path.append('/usr/local/lib/python3.9/site-packages')
```



Wenn die Zeile

```
!unzip /mydrive/freedomtech.zip
```

durchgelaufen ist, muss die train.py Datei hochgeladen werden.



1. Auf das Dateisymbol klicken
2. Auf Hochladen klicken
3. train.py auswählen

Als nächstes klicken wir links im Google Colab unsere train.py an woraufhin sich auf der rechten Seite ein python Skript öffnet.

Hier müssen wir in Zeile 22 und Zeile 28 die Label eintragen, welche wir mit der Labelling erstellt haben (z.B. wie bei uns USB und Presenter).

```
19 train_data = object_detector.DataLoader.from_pascal_voc(  
20     'freedomtech/train',  
21     'freedomtech/train',  
22     ['esp8266', 'pico']  
23 )  
24  
25 val_data = object_detector.DataLoader.from_pascal_voc(  
26     'freedomtech/validate',  
27     'freedomtech/validate',  
28     ['esp8266', 'pico']
```

Hier die eigenen Label eintragen

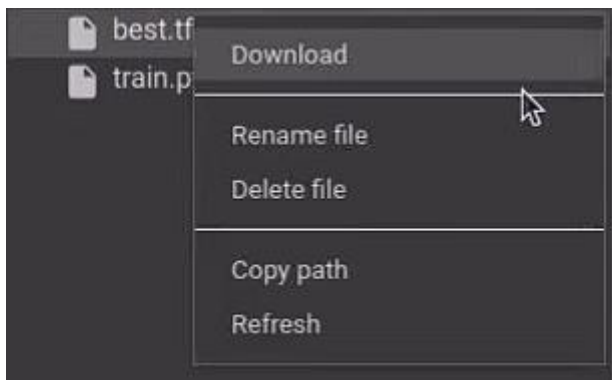
Außerdem ändern wir in Zeile 40 den epochs Wert von 20 auf 100

```
39
40 _size=4, train_whole_model=True, epochs=100,
41
```

Wir speichern unsere train.py Datei schließen diese und führen den letzten Satz aus.

Dieser Schritt dauert sehr lange, abhängig davon wie viele Bilder/Objekte erstellt wurden. Am besten über Nacht laufen lassen.

Google Colab erstellt uns nun daraus eine best.tflite Datei, welche wir auf der linken Seite herunterladen können.



Diese Datei kopieren wir in unseren tflite Ordner.

Nun müssen wir nur noch kontrollieren, dass die korrekte Datei aktiv ist und unsere eigene angelernte Objekterkennung sollte funktionieren.

```
132 # default='efficientdet_lite0.tflite')
133 default='best.tflite')
```

## 5 Nachwort

Das Projekt hat uns durch die vielen Kommunikationsproblemen zwischen den einzelnen Komponenten teilweise sehr verzweifeln lassen. Dafür war der Augenblick in dem von dem Raspberry der erste Schraubenzieher als Zahnbürste erkannt wurde umso schöner.

Auf die Integration in ein LOGO Netzwerk wird an dieser Stelle nicht eingegangen, die daran angepasste tflite Datei kann jedoch unter

[https://github.com/f2tv23ki/bbz\\_ki](https://github.com/f2tv23ki/bbz_ki)

heruntergeladen werden.

Wir hoffen, dass das Dokument als hilfreicher Lernfaden dient und wünschen allen zukünftigen Nutzern viel Spaß und Erfolg bei ihren Projekten.