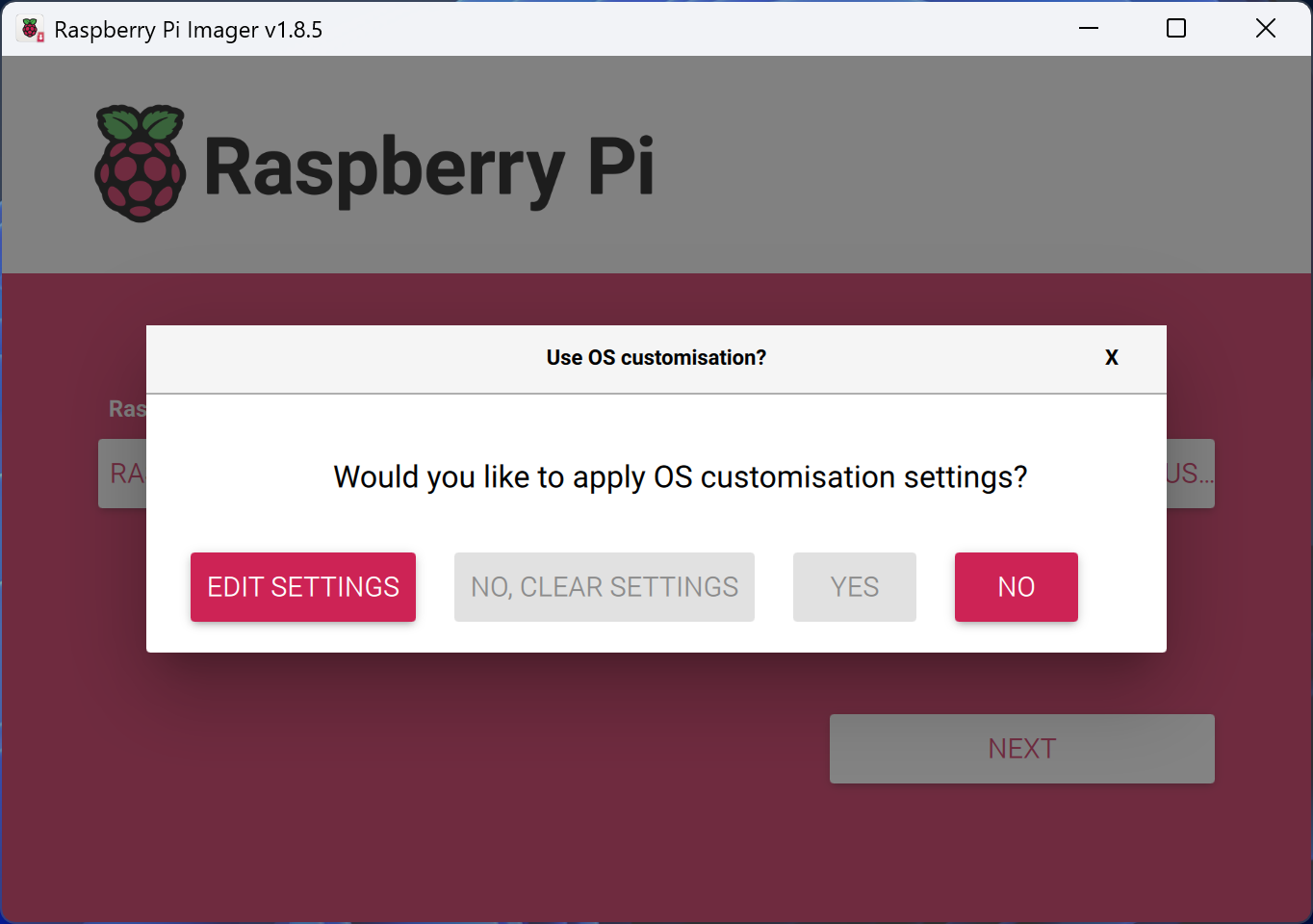
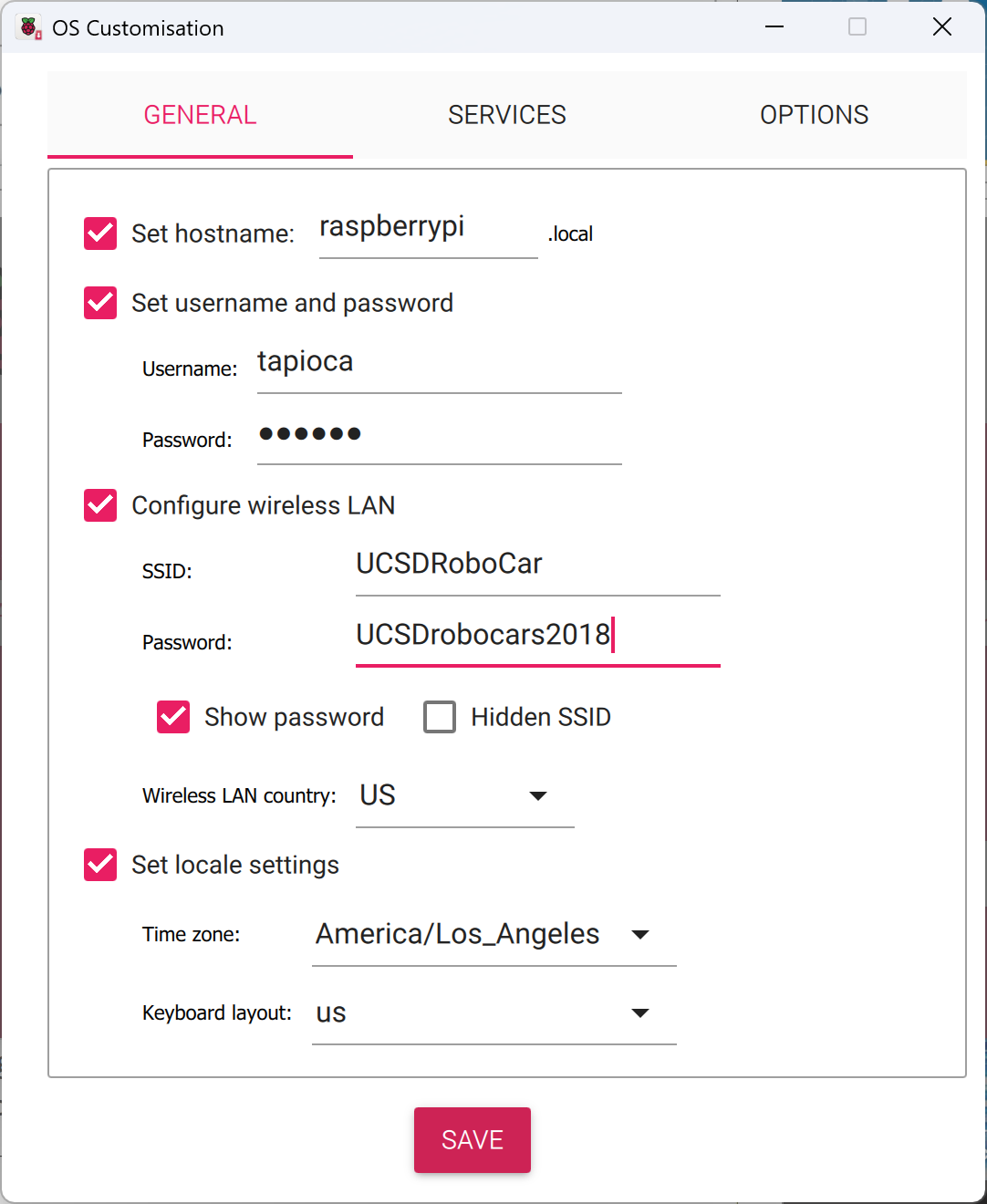
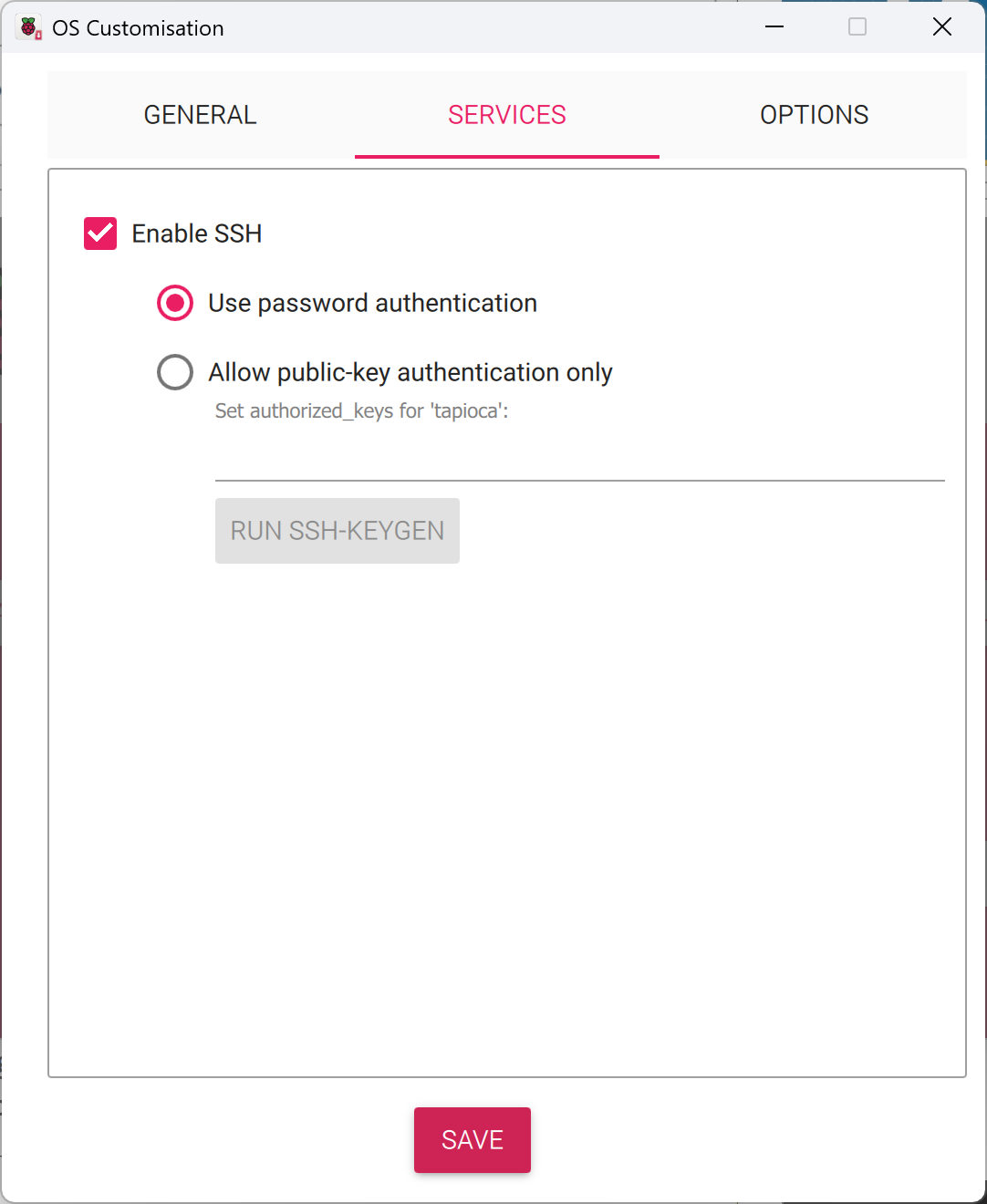
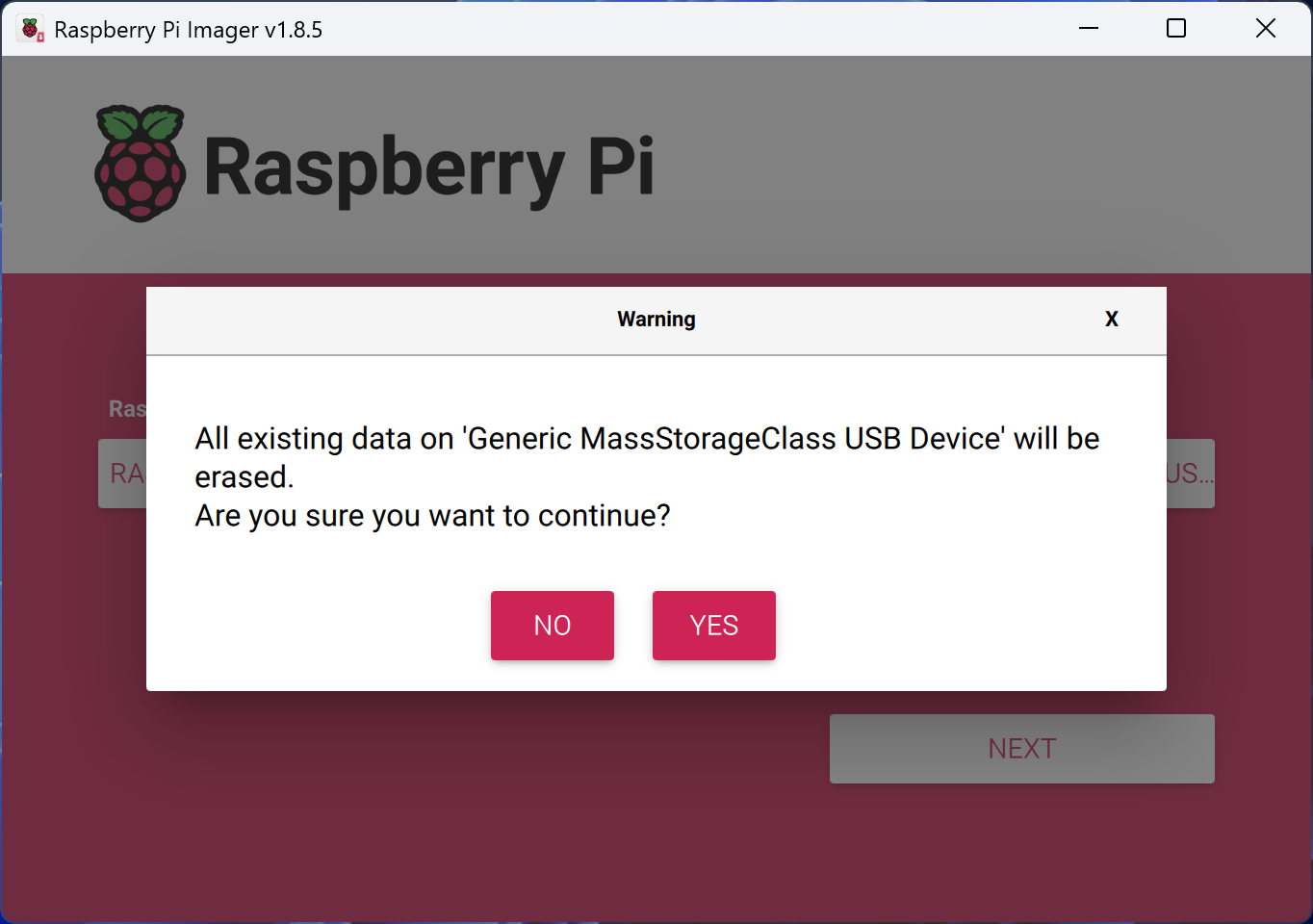
**Raspberry Pi Donkey Setup**

**1. Setting Up the Raspberry Pi**

a. Download Raspberry Pi Imager and run: <https://www.raspberrypi.com/software/>

b. Edit settings  


c. In the General Tab: Set Username, Password, and Setup Wifi  


d. In the Services Tab: Enable SSH with Password, and hit save  
  
e. Allow SD card formatting  
  
f. SSH into the Raspberry Pi:

| ssh raspberpi\_name@raspberrypi.local |
| --- |

g. Update Raspberry Pi:

| sudo apt update && sudo apt full-upgrade sudo rpi-eeprom-update -a |
| --- |

**2. Setting Up DonkeyCar (5.1) on Raspberry Pi**

a. Update and Upgrade

| sudo apt-get update --allow-releaseinfo-change sudo apt-get upgrade |
| --- |

b. Configure Raspberry Pi

| sudo raspi-config |
| --- |

- Enable Interfacing Options - I2C

- Select Advanced Options - Expand Filesystem so you can use your whole SD-card storage

- Do not enable the legacy camera (it's disabled by default, so don't change anything)

- Choose Finish and Rebbot Raspberry Pi

c. Setup Virtual Environment

| python3 -m venv donkey --system-site-packages echo "source ~/donkey/bin/activate" >> ~/.bashrc source ~/.bashrc |
| --- |

d. Install Required Libraries and Donkey Car

| sudo apt install libcap-dev libhdf5-dev libhdf5-serial-dev  pip install donkeycar[pi] |
| --- |

e. Setup Donkeycar directory

| mkdir projects cd projects git clone https://github.com/autorope/donkeycar cd donkeycar git checkout main pip install -e .[pi] |
| --- |

f. Make Car (d4 is the name of the car and can be changed).

| cd ~/projects/donkeycar donkey createcar --path ~/projects/d4 |
| --- |

**3. Benchmarking Raspberry Pi with only Tensorflow**

a. Locate profile.py

| Cd  source donkey/bin/activate  find ~/ -name "profile.py" |
| --- |

Expected output: Path to profile.py (~/projects/donkeycar/scripts/profile.py)

b. Create a copy in the car folder (d4)

| cp ~/projects/donkeycar/script/profile.py ~/projects/d4/ |
| --- |

c. Run profile.py to test the model’s fps (have your model saved as an .h5 file in d4/models)

| cd projects/d4 python profile.py --model=./models/modelname.h5 --type=linear |
| --- |

**4. Benchmarking Raspberry Pi with only Tensorflow Lite**

a. Convert .h5 to .tflite

| cd ~/projects/donkeycar/scripts python tflite\_convert --model=~/projects/d4/models/{modelname}.h5 --out=~/projects/d4/models/{modelname}.tflite |
| --- |

b. Locate tflite\_profile.py

| Cd  source donkey/bin/activate  find ~/ -name "tflite\_profile.py" |
| --- |

Expected output: Path to profile.py (~/projects/donkeycar/scripts/tflite\_profile.py)

b. Create a copy in the car folder (d4)

| cp ~/projects/donkeycar/script/tflite\_profile.py ~/projects/d4/ |
| --- |

c. Run profile.py to test the model’s fps (have your model saved as an .tflite file in d4/models)

| cd projects/d4 python profile.py --model=./models/modelname.tflite |
| --- |

**5. Setting up Hailo on the Raspberry Pi**

a. Install Hailo AI Hat onto the Raspberry Pi and connect the ribbon connector

b. Install AI Hat on the Raspberry Pi

| sudo apt install hailo-all |
| --- |

c. Restart Raspberry Pi

| sudo restart now |
| --- |

d. Verify Installation

| hailortcli fw-control identify |
| --- |

Expected Result:

| Executing on device: 0000:01:00.0 Identifying board Control Protocol Version: 2 Firmware Version: 4.17.0 (release,app,extended context switch buffer) Logger Version: 0 Board Name: Hailo-8 Device Architecture: HAILO8L Serial Number: N/A Part Number: N/A Product Name: N/A |
| --- |

**5. Converting from Tensorflow to .hef**

Just like a Jetson Nano uses TensorRT to accelerate the model using the GPU, the Raspberry Pi uses Hailo RT to optimize and use its chip. This model can only work with Hailo Executible File (.hef) models, thus posing the need to convert.

Converting to .hef requires using a Linux computer (or an instance of WSL) and cannot be done on the Raspberry Pi.

Requirements:  
Ubuntu 20.04/22.04, 64-bit (supported also on Windows, under WSL2)

16+ GB RAM (32+ GB recommended)

Python 3.8/3.9/3.10, including pip and virtualenv

python3.X-dev and python3.X-distutils (according to the Python version), python3-tk, graphviz, and libgraphviz-dev packages. Use the command sudo apt-get install PACKAGE for installation.

.tflite model (Refer to section 4.a. To convert .h5 to .tflite).

The following additional requirements are needed for GPU based hardware emulation:

Nvidia’s Pascal/Turing/Ampere GPU architecture (such as Titan X Pascal, GTX 1080 Ti, RTX 2080 Ti, or RTX A4000)

GPU driver version 525 or higher

CUDA 11.8 or higher

CUDNN 8.9

a. Installing WSL: In your command line: (Provide a username and password). Run line by line

| wsl --list --online wsl --install -d Ubuntu-22.04 sudo apt update |
| --- |

b. Install Dependencies

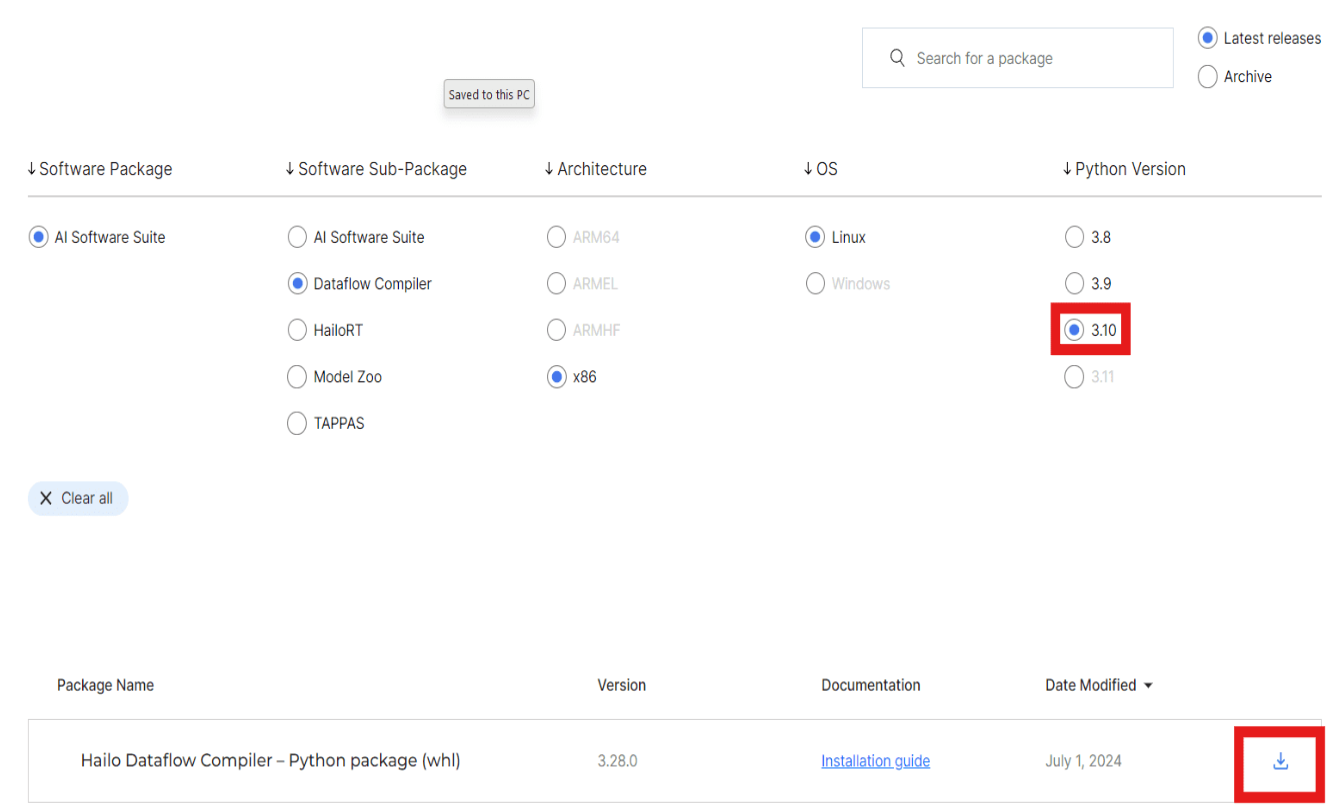
| sudo apt install python3-pip sudo apt install python3.10-venv sudo apt-get install python3.10-dev python3.10-distutils python3-tk libfuse2 graphviz libgraphviz-dev sudo pip install pygraphviz sudo apt install wslu |
| --- |

c. Create Virtual Environment

| python3 -m venv hailodfc . hailodfc/bin/activate |
| --- |

d. Check Python Version and Download [Dataflow Compiler Package](https://hailo.ai/developer-zone/software-downloads/) (sign-up required):

| python3 --version |
| --- |



e. Go to WSL view and move the file to your <username> directory

| wslview . |
| --- |

f. Install Hailo DFC (change command based on your file name):

| pip3 install hailo\_dataflow\_compiler-3.28.0-py3-none-linux\_x86\_64.whl |
| --- |

g. Verify Install (ignore warnings if pre requisites are met):

| hailo -h |
| --- |

h. Create models directory:

| mkdir models cd models |
| --- |

i. Move the .tfllite model to the models directory using WSL View:

| wslview . |
| --- |

j. Convert .tflite file to Hailo Archive Formart (.har):

| nano parser.py |
| --- |

k. Paste the following code in parser.py and save (ctrl +s) and close (ctrl+x):

| # General imports used throughout the tutorial import tensorflow as tf from IPython.display import SVG # import the ClientRunner class from the hailo\_sdk\_client package from hailo\_sdk\_client import ClientRunner  chosen\_hw\_arch = "hailo8"  model\_name = "<model name>" model\_path = "<model path from models folder>" runner = ClientRunner(hw\_arch=chosen\_hw\_arch) hn, npz = runner.translate\_tf\_model(model\_path, model\_name)  hailo\_model\_har\_name = f"{model\_name}\_hailo\_model.har" runner.save\_har(hailo\_model\_har\_name) |
| --- |

l. Run Parser.py:

| python parser.py |
| --- |

m. Optimize the .har model:

| nano optimizer.py |
| --- |

n. Paste the following code in optimizer.py and save (change image resolution in preproc and calibdataset):

| import json import os import numpy as np import tensorflow as tf from IPython.display import SVG from matplotlib import patches from matplotlib import pyplot as plt from PIL import Image from tensorflow.python.eager.context import eager\_mode from hailo\_sdk\_client import ClientRunner, InferenceContext IMAGES\_TO\_VISUALIZE = 100 #only have these many images in your folder   # First, we will prepare the calibration set. Resize the images to the correct size and crop them. def preproc(image, output\_height=120, output\_width=160, resize\_side=160):  with eager\_mode():  h, w = image.shape[0], image.shape[1]  scale = tf.cond(tf.less(h, w), lambda: resize\_side / h, lambda: resize\_side / w)  resized\_image = tf.compat.v1.image.resize\_bilinear(tf.expand\_dims(image, 0), [int(h \* scale), int(w \* scale)])  cropped\_image = tf.compat.v1.image.resize\_with\_crop\_or\_pad(resized\_image, output\_height, output\_width)  return tf.squeeze(cropped\_image)  images\_path = "<Path to Images>" images\_list = [img\_name for img\_name in os.listdir(images\_path) if os.path.splitext(img\_name)[1] == ".jpg"] calib\_dataset = np.zeros((len(images\_list), 120, 160, 3)) for idx, img\_name in enumerate(sorted(images\_list)):  img = np.array(Image.open(os.path.join(images\_path, img\_name)))  img\_preproc = preproc(img)  calib\_dataset[idx, :, :, :] = img\_preproc.numpy() np.save("calib\_set.npy", calib\_dataset) model\_name = "<modelname>" hailo\_model\_har\_name = f"{model\_name}\_hailo\_model.har" assert os.path.isfile(hailo\_model\_har\_name), "Please provide valid path for HAR file" runner = ClientRunner(har=hailo\_model\_har\_name)  alls = "normalization1 = normalization([123.675, 116.28, 103.53], [58.395, 57.12, 57.375])\n"  runner.load\_model\_script(alls) # Call Optimize to perform the optimization process runner.optimize(calib\_dataset) # Save the result state to a Quantized HAR file quantized\_model\_har\_path = f"{model\_name}\_quantized\_model.har" runner.save\_har(quantized\_model\_har\_path) |
| --- |

o. Run optimizer.py:

| python omptimizer.py |
| --- |

p. Compile to .hef by creating a compiler.py:

| nano compiler.py |
| --- |

q. Paste the following code in compiler.py and save:

| from hailo\_sdk\_client import ClientRunner  model\_name = "<modelname>" quantized\_model\_har\_path = f"{model\_name}\_quantized\_model.har" runner = ClientRunner(har=quantized\_model\_har\_path)  hef = runner.compile() file\_name = f"{model\_name}.hef" with open(file\_name, "wb") as f:  f.write(hef) |
| --- |

r. Run Compiler.py

| python compiler.py |
| --- |

s. Copy the .hef file from WSL directory using wslview and scp to raspberry pi

To copy: wslview .

To scp from the command line:

| scp <file path on computer> <rpi name>@raspberrypi:~/projects/d4/models |
| --- |

**6. Benchmarking the Hailo AI Hat**a. SSH into the Raspberry Pi and cd to the models directory:

| cd projects/d4/models |
| --- |

b. Benchmark with AI Hat:

| hailortcli benchmark <model\_name>.hef |
| --- |