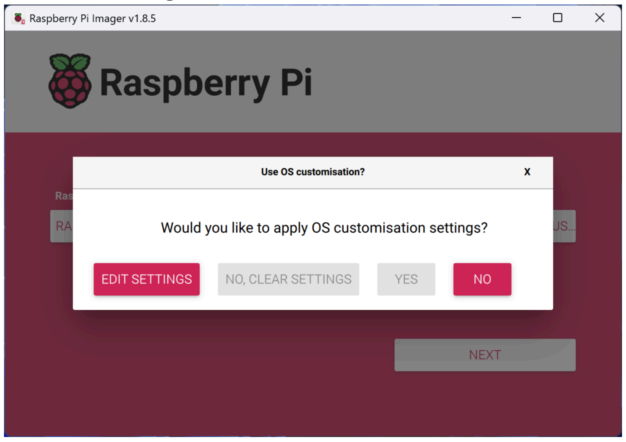
Raspberry Pi Donkey Setup

1. Setting Up the Raspberry Pi

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

b. Edit settings

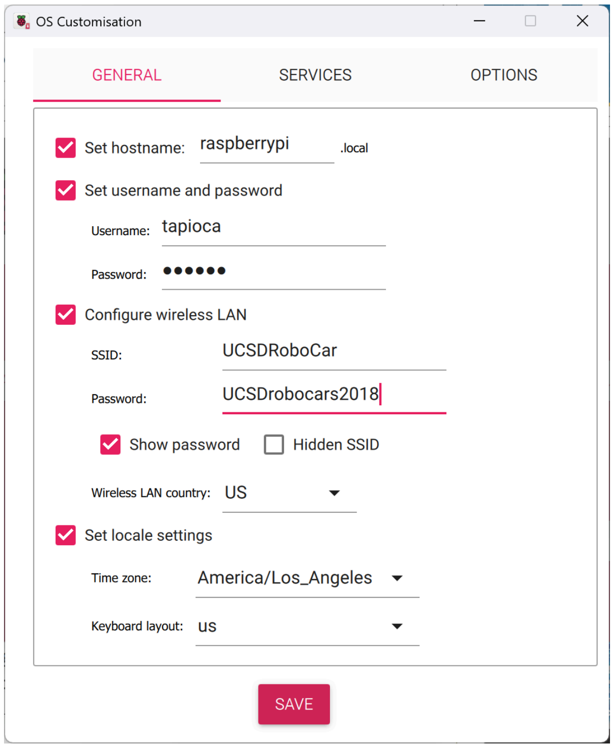


Raspberry Pi Imager v1.8.5

Use OS customisation?

Would you like to apply OS customisation settings?

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



OS Customisation

☑ Set hostname: raspberrypi .local

☑ Set username and password

Username: tapioca

Password: ●●●●●●

☑ Configure wireless LAN

SSID: UCSDRoboCar

Password: UCSDrobocars2018

☑ Show password ☐ Hidden SSID

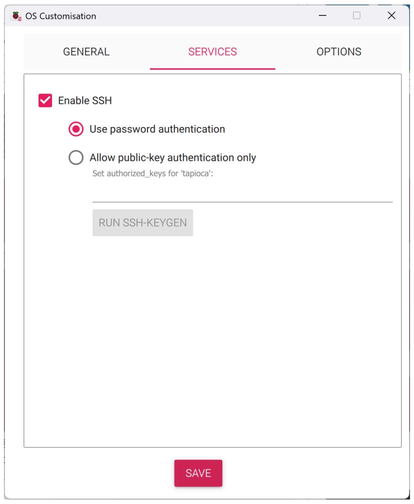
Wireless LAN country: US

☑ Set locale settings

Time zone: America/Los_Angeles

Keyboard layout: us

d. In the Services Tab: Enable SSH with Password, and hit save


OS Customisation

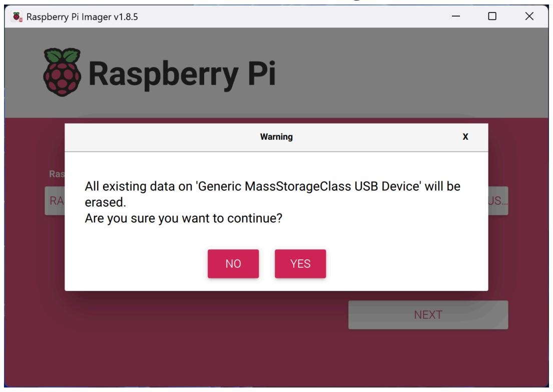
[✓] Enable SSH

[● Use password authentication]

Allow public-key authentication only

Set authorized_keys for 'tapioca':

e. Allow SD card formatting


Raspberry Pi Imager v1.8.5

Warning

All existing data on 'Generic MassStorageClass USB Device' will be erased.
Are you sure you want to continue?

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

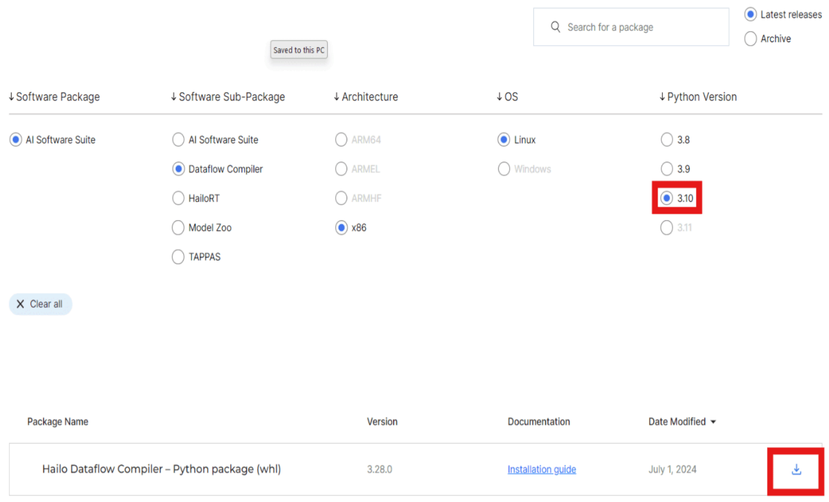
```
sudo apt install python3-pip
sudo apt install python3.10-venv
sudo apt-get install python3.10-dev python3.10-distutils python3-tk
libfuse2 graphviz libgraphviz-dev
sudo pip install pygraphviz
sudo apt install wslu
```

c. Create Virtual Environment

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

d. Check Python Version and Download Dataflow Compiler Package (sign-up required):

```
python3 --version
```



[✓ Latest releases] Archive

↓Software Package	↓Software Sub-Package	↓Architecture	↓OS	↓Python Version
[✓ AI Software Suite]	AI Software Suite	ARM64	[✓ Linux]	3.8
	[✓ Dataflow Compiler]	ARMEL	Windows	3.9
	HailoRT	ARMHF		[✓ 3.10]
	Model Zoo	[✓ x86]		3.11
	TAPPAS			

Package Name	Version	Documentation	Date Modified
Hailo Dataflow Compiler – Python package (whl)	3.28.0	Installation guide	July 1, 2024

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
```