# 充电口识别及立体空间坐标计算V1.0

# 源代码

#主代码Start

#导入库文件

from machine import I2C

from vl53l1x import VL53L1X

import sensor, image, time, os, tf, math, uos, gc, pyb ,json ,ustruct

from pyb import UART

#设置相机基础参数

sensor.reset()

sensor.set\_pixformat(sensor.RGB565)

sensor.set\_framesize(sensor.QVGA)

sensor.set\_windowing((240, 240))

sensor.skip\_frames(time=2000)

net = None

labels = None

min\_confidence = 0.5

i2c = I2C(2)

distance = VL53L1X(i2c)

uart = UART(1,9600)

uart.init(9600, bits=8, parity=None, stop=1)

#打包函数，用于将数据打包发出，便于接收

def send\_data\_packet(x, y, z):

temp = ustruct.pack("<bbhhh",

0x12,

0x2C,

int(x),

int(y),

int(z))

uart.write(temp)

#指示灯设置

ledr = pyb.LED(1)

ledg = pyb.LED(2)

#模型运行

K = 0.16

try:

net = tf.load("trained.tflite", load\_to\_fb=uos.stat('trained.tflite')[6] > (gc.mem\_free() - (64\*1024)))

except Exception as e:

raise Exception('Failed to load "trained.tflite", did you copy the .tflite and labels.txt file onto the mass-storage device? (' + str(e) + ')')

try:

labels = [line.rstrip('\n') for line in open("labels.txt")]

except Exception as e:

raise Exception('Failed to load "labels.txt", did you copy the .tflite and labels.txt file onto the mass-storage device? (' + str(e) + ')')

colors = [

(255, 0, 0),

( 0, 255, 0),

(255, 255, 0),

( 0, 0, 255),

(255, 0, 255),

( 0, 255, 255),

(255, 255, 255),

]

running = True

clock = time.clock()

run\_model = True

#通信主体函数

def chuli\_data(x, y, z):

# 检查UART是否有数据可读

if uart.any():

# 读取UART数据并解码

a = uart.read().decode()

print(a)

# 根据接收到的数据执行不同的操作

if a == "ok":

# 打开LED灯（假设ledg是一个LED对象）

ledg.on()

# 如果模型正在运行，则发送数据包

if run\_model:

send\_data\_packet(x, y, z)

# 等待2000毫秒（2秒）

time.sleep\_ms(2000)

elif a == "no":

# 如果接收到"no"，则停止模型运行并点亮红色LED（假设ledr是一个LED对象）

run\_model = False

ledr.on()

# 向UART发送一个字节0xFF

uart.write('0xff')

elif a == "start":

# 如果接收到"start"，则开始模型运行

run\_model = True

#循环主体

while running:

clock.tick() # 更新时钟

img = sensor.snapshot() # 获取图像快照

# 遍历每个检测对象及其列表

for i, detection\_list in enumerate(net.detect(img, thresholds=[(math.ceil(min\_confidence \* 255), 255)])):

if i == 0:

continue # 跳过第一个检测对象（可能是背景或不需要检测的对象）

if len(detection\_list) == 0:

continue # 如果当前检测列表为空，则跳过

# 打印检测对象的标签

print("\*\*\*\*\*\*\*\*\*\* %s \*\*\*\*\*\*\*\*\*\*" % labels[i])

# 遍历每个检测到的对象

for d in detection\_list:

x, y, w, h = d.rect() # 获取对象的边界框

center\_x = math.floor(x + (w / 2)) # 计算中心点的 x 坐标

center\_y = math.floor(y + (h / 2)) # 计算中心点的 y 坐标

# 打印中心点坐标

print('x %d\ty %d' % (center\_x, center\_y))

# 在图像上绘制一个圆来表示检测到的对象

img.draw\_circle((center\_x, center\_y, 12), color=colors[i], thickness=2)

# 计算 X, Y, Z 值

X = center\_x \* K \* 10

Y = distance.read() # 读取距离值作为 Y

Z = (-center\_y + 240) \* K \* 10 # 根据给定的公式计算 Z 值

# 打印范围值

print("rangex: mm ", X)

print("rangey: mm ", Y)

print("rangez: mm ", Z)

# 调用 chuli\_data 函数并传递 X, Y, Z 值

chuli\_data(X, Y, Z)

# 打印当前帧率

print(clock.fps(), "fps", end="\n\n")

#主代码end

#库函数Start

#测距库

import time

VL51L1X\_DEFAULT\_CONFIGURATION = bytes(

[

0x00, # 0x2d : set bit 2 and 5 to 1 for fast plus mode (1MHz I2C), else don't touch

0x00, # 0x2e : bit 0 if I2C pulled up at 1.8V, else set bit 0 to 1 (pull up at AVDD)

0x00, # 0x2f : bit 0 if GPIO pulled up at 1.8V, else set bit 0 to 1 (pull up at AVDD)

0x01, # 0x30 : set bit 4 to 0 for active high interrupt and 1 for active low

# (bits 3:0 must be 0x1), use SetInterruptPolarity()

0x02, # 0x31 : bit 1 = interrupt depending on the polarity, use CheckForDataReady()

0x00, # 0x32 : not user-modifiable

0x02, # 0x33 : not user-modifiable

0x08, # 0x34 : not user-modifiable

0x00, # 0x35 : not user-modifiable

0x08, # 0x36 : not user-modifiable

0x10, # 0x37 : not user-modifiable

0x01, # 0x38 : not user-modifiable

0x01, # 0x39 : not user-modifiable

0x00, # 0x3a : not user-modifiable

0x00, # 0x3b : not user-modifiable

0x00, # 0x3c : not user-modifiable

0x00, # 0x3d : not user-modifiable

0xFF, # 0x3e : not user-modifiable

0x00, # 0x3f : not user-modifiable

0x0F, # 0x40 : not user-modifiable

0x00, # 0x41 : not user-modifiable

0x00, # 0x42 : not user-modifiable

0x00, # 0x43 : not user-modifiable

0x00, # 0x44 : not user-modifiable

0x00, # 0x45 : not user-modifiable

0x20, # 0x46 : interrupt configuration 0->level low detection,

# 1-> level high, 2-> Out of window, 3->In window, 0x20-> New sample ready , TBC

0x0B, # 0x47 : not user-modifiable

0x00, # 0x48 : not user-modifiable

0x00, # 0x49 : not user-modifiable

0x02, # 0x4a : not user-modifiable

0x0A, # 0x4b : not user-modifiable

0x21, # 0x4c : not user-modifiable

0x00, # 0x4d : not user-modifiable

0x00, # 0x4e : not user-modifiable

0x05, # 0x4f : not user-modifiable

0x00, # 0x50 : not user-modifiable

0x00, # 0x51 : not user-modifiable

0x00, # 0x52 : not user-modifiable

0x00, # 0x53 : not user-modifiable

0xC8, # 0x54 : not user-modifiable

0x00, # 0x55 : not user-modifiable

0x00, # 0x56 : not user-modifiable

0x38, # 0x57 : not user-modifiable

0xFF, # 0x58 : not user-modifiable

0x01, # 0x59 : not user-modifiable

0x00, # 0x5a : not user-modifiable

0x08, # 0x5b : not user-modifiable

0x00, # 0x5c : not user-modifiable

0x00, # 0x5d : not user-modifiable

0x01, # 0x5e : not user-modifiable

0xDB, # 0x5f : not user-modifiable

0x0F, # 0x60 : not user-modifiable

0x01, # 0x61 : not user-modifiable

0xF1, # 0x62 : not user-modifiable

0x0D, # 0x63 : not user-modifiable

0x01, # 0x64 : Sigma threshold MSB (mm in 14.2 format for MSB+LSB),

# use SetSigmaThreshold(), default value 90 mm

0x68, # 0x65 : Sigma threshold LSB

0x00, # 0x66 : Min count Rate MSB (MCPS in 9.7 format for MSB+LSB), use SetSignalThreshold()

0x80, # 0x67 : Min count Rate LSB

0x08, # 0x68 : not user-modifiable

0xB8, # 0x69 : not user-modifiable

0x00, # 0x6a : not user-modifiable

0x00, # 0x6b : not user-modifiable

0x00, # 0x6c : Intermeasurement period MSB, 32 bits register, use SetIntermeasurementInMs()

0x00, # 0x6d : Intermeasurement period

0x0F, # 0x6e : Intermeasurement period

0x89, # 0x6f : Intermeasurement period LSB

0x00, # 0x70 : not user-modifiable

0x00, # 0x71 : not user-modifiable

0x00, # 0x72 : distance threshold high MSB (in mm, MSB+LSB), use SetD:tanceThreshold()

0x00, # 0x73 : distance threshold high LSB

0x00, # 0x74 : distance threshold low MSB ( in mm, MSB+LSB), use SetD:tanceThreshold()

0x00, # 0x75 : distance threshold low LSB

0x00, # 0x76 : not user-modifiable

0x01, # 0x77 : not user-modifiable

0x0F, # 0x78 : not user-modifiable

0x0D, # 0x79 : not user-modifiable

0x0E, # 0x7a : not user-modifiable

0x0E, # 0x7b : not user-modifiable

0x00, # 0x7c : not user-modifiable

0x00, # 0x7d : not user-modifiable

0x02, # 0x7e : not user-modifiable

0xC7, # 0x7f : ROI center, use SetROI()

0xFF, # 0x80 : XY ROI (X=Width, Y=Height), use SetROI()

0x9B, # 0x81 : not user-modifiable

0x00, # 0x82 : not user-modifiable

0x00, # 0x83 : not user-modifiable

0x00, # 0x84 : not user-modifiable

0x01, # 0x85 : not user-modifiable

0x01, # 0x86 : clear interrupt, use ClearInterrupt()

0x40, # 0x87 : start ranging, use StartRanging() or StopRanging(), If you want

# an automatic start after VL53L1X\_init() call, put 0x40 in location 0x87

]

)

class VL53L1X:

def \_\_init\_\_(self, i2c, address=0x29):

self.i2c = i2c

self.address = address

self.reset()

time.sleep\_ms(1)

if self.read\_model\_id() != 0xEACC:

raise RuntimeError("Failed to find expected ID register values. Check wiring!")

# write default configuration

self.i2c.writeto\_mem(self.address, 0x2D, VL51L1X\_DEFAULT\_CONFIGURATION, addrsize=16)

# the API triggers this change in VL53L1\_init\_and\_start\_range() once a

# measurement is started; assumes MM1 and MM2 are disabled

self.writeReg16Bit(0x001E, self.readReg16Bit(0x0022) \* 4)

time.sleep\_ms(200)

def writeReg(self, reg, value):

return self.i2c.writeto\_mem(self.address, reg, bytes([value]), addrsize=16)

def writeReg16Bit(self, reg, value):

return self.i2c.writeto\_mem(

self.address, reg, bytes([(value >> 8) & 0xFF, value & 0xFF]), addrsize=16

)

def readReg(self, reg):

return self.i2c.readfrom\_mem(self.address, reg, 1, addrsize=16)[0]

def readReg16Bit(self, reg):

data = self.i2c.readfrom\_mem(self.address, reg, 2, addrsize=16)

return (data[0] << 8) + data[1]

def read\_model\_id(self):

return self.readReg16Bit(0x010F)

def reset(self):

self.writeReg(0x0000, 0x00)

time.sleep\_ms(100)

self.writeReg(0x0000, 0x01)

def read(self):

data = self.i2c.readfrom\_mem(self.address, 0x0089, 17, addrsize=16) # RESULT\_\_RANGE\_STATUS

# range\_status = data[0]

# report\_status = data[1]

# stream\_count = data[2]

# dss\_actual\_effective\_spads\_sd0 = (data[3] << 8) + data[4]

# peak\_signal\_count\_rate\_mcps\_sd0 = (data[5]<<8) + data[6]

# ambient\_count\_rate\_mcps\_sd0 = (data[7] << 8) + data[8]

# sigma\_sd0 = (data[9]<<8) + data[10]

# phase\_sd0 = (data[11]<<8) + data[12]

final\_crosstalk\_corrected\_range\_mm\_sd0 = (data[13] << 8) + data[14]

# peak\_signal\_count\_rate\_crosstalk\_corrected\_mcps\_sd0 = (data[15] << 8) + data[16]

# status = None

# if range\_status in (17, 2, 1, 3):

# status = "HardwareFail"

# elif range\_status == 13:

# status = "MinRangeFail"

# elif range\_status == 18:

# status = "SynchronizationInt"

# elif range\_status == 5:

# status = "OutOfBoundsFail"

# elif range\_status == 4:

# status = "SignalFail"

# elif range\_status == 6:

# status = "SignalFail"

# elif range\_status == 7:

# status = "WrapTargetFail"

# elif range\_status == 12:

# status = "XtalkSignalFail"

# elif range\_status == 8:

# status = "RangeValidMinRangeClipped"

# elif range\_status == 9:

# if stream\_count == 0:

# status = "RangeValidNoWrapCheckFail"

# else:

# status = "OK"

return final\_crosstalk\_corrected\_range\_mm\_sd0

#pyb库

#functions related to the board

module:: pyb

:synopsis: functions related to the board

The ``pyb`` module contains specific functions related to the board.

Time related functions

#以下为功能代码的定义部分

.. function:: delay(ms)

Delay for the given number of milliseconds.

.. function:: udelay(us)

Delay for the given number of microseconds.

.. function:: millis()

Returns the number of milliseconds since the board was last reset.

The result is always a MicroPython smallint (31-bit signed number), so

after 2^30 milliseconds (about 12.4 days) this will start to return

negative numbers.

Note that if :meth:`pyb.stop()` is issued the hardware counter supporting this

function will pause for the duration of the "sleeping" state. This

will affect the outcome of :meth:`pyb.elapsed\_millis()`.

.. function:: micros()

Returns the number of microseconds since the board was last reset.

The result is always a MicroPython smallint (31-bit signed number), so

after 2^30 microseconds (about 17.8 minutes) this will start to return

negative numbers.

Note that if :meth:`pyb.stop()` is issued the hardware counter supporting this

function will pause for the duration of the "sleeping" state. This

will affect the outcome of :meth:`pyb.elapsed\_micros()`.

.. function:: elapsed\_millis(start)

Returns the number of milliseconds which have elapsed since ``start``.

This function takes care of counter wrap, and always returns a positive

number. This means it can be used to measure periods up to about 12.4 days.

Example::

start = pyb.millis()

while pyb.elapsed\_millis(start) < 1000:

# Perform some operation

.. function:: elapsed\_micros(start)

Returns the number of microseconds which have elapsed since ``start``.

This function takes care of counter wrap, and always returns a positive

number. This means it can be used to measure periods up to about 17.8 minutes.

Example::

start = pyb.micros()

while pyb.elapsed\_micros(start) < 1000:

# Perform some operation

pass

Reset related functions

-----------------------

.. function:: hard\_reset()

Resets the OpenMV Cam in a manner similar to pushing the external RESET

button.

.. function:: bootloader()

Activate the bootloader without BOOT\\* pins.

.. function:: fault\_debug(value)

Enable or disable hard-fault debugging. A hard-fault is when there is a fatal

error in the underlying system, like an invalid memory access.

If the \*value\* argument is ``False`` then the board will automatically reset if

there is a hard fault.

If \*value\* is ``True`` then, when the board has a hard fault, it will print the

registers and the stack trace, and then cycle the LEDs indefinitely.

The default value is disabled, i.e. to automatically reset.

Interrupt related functions

---------------------------

.. function:: disable\_irq()

Disable interrupt requests.

Returns the previous IRQ state: ``False``/``True`` for disabled/enabled IRQs

respectively. This return value can be passed to enable\_irq to restore

the IRQ to its original state.

.. function:: enable\_irq(state=True)

Enable interrupt requests.

If ``state`` is ``True`` (the default value) then IRQs are enabled.

If ``state`` is ``False`` then IRQs are disabled. The most common use of

this function is to pass it the value returned by ``disable\_irq`` to

exit a critical section.

Power related functions

-----------------------

.. function:: wfi()

Wait for an internal or external interrupt.

This executes a ``wfi`` instruction which reduces power consumption

of the MCU until any interrupt occurs (be it internal or external),

at which point execution continues. Note that the system-tick interrupt

occurs once every millisecond (1000Hz) so this function will block for

at most 1ms.

.. function:: stop()

Put the OpenMV Cam in a "sleeping" state.

This reduces power consumption to less than 500 uA. To wake from this

sleep state requires an external interrupt or a real-time-clock event.

Upon waking execution continues where it left off.

See :meth:`rtc.wakeup` to configure a real-time-clock wakeup event.

.. function:: standby()

Put the OpenMV Cam into a "deep sleep" state.

This reduces power consumption to less than 50 uA. To wake from this

sleep state requires a real-time-clock event.

Upon waking the system undergoes a hard reset.

See :meth:`rtc.wakeup` to configure a real-time-clock wakeup event.

Miscellaneous functions

-----------------------

.. function:: have\_cdc()

Return True if USB is connected as a serial device, False otherwise.

.. note:: This function is deprecated. Use pyb.USB\_VCP().isconnected() instead.

.. function:: hid((buttons, x, y, z))

Takes a 4-tuple (or list) and sends it to the USB host (the PC) to

signal a HID mouse-motion event.

.. note:: This function is deprecated. Use :meth:`pyb.USB\_HID.send()` instead.

.. function:: info([dump\_alloc\_table])

Print out lots of information about the board.

.. function:: main(filename)

Set the filename of the main script to run after boot.py is finished. If

this function is not called then the default file main.py will be executed.

It only makes sense to call this function from within boot.py.

.. function:: mount(device, mountpoint, \*, readonly=False, mkfs=False)

.. note:: This function is deprecated. Mounting and unmounting devices should

be performed by :meth:`os.mount` and :meth:`os.umount` instead.

Mount a block device and make it available as part of the filesystem.

``device`` must be an object that provides the block protocol. (The

following is also deprecated. See :class:`os.AbstractBlockDev` for the

correct way to create a block device.)

- ``readblocks(self, blocknum, buf)``

- ``writeblocks(self, blocknum, buf)`` (optional)

- ``count(self)``

- ``sync(self)`` (optional)

``readblocks`` and ``writeblocks`` should copy data between ``buf`` and

the block device, starting from block number ``blocknum`` on the device.

``buf`` will be a bytearray with length a multiple of 512. If

``writeblocks`` is not defined then the device is mounted read-only.

The return value of these two functions is ignored.

``count`` should return the number of blocks available on the device.

``sync``, if implemented, should sync the data on the device.

The parameter ``mountpoint`` is the location in the root of the filesystem

to mount the device. It must begin with a forward-slash.

If ``readonly`` is ``True``, then the device is mounted read-only,

otherwise it is mounted read-write.

If ``mkfs`` is ``True``, then a new filesystem is created if one does not

already exist.

.. function:: repl\_uart(uart)

Get or set the UART object where the REPL is repeated on.

. function:: rng()

Return a 30-bit hardware generated random number.

.. function:: sync()

Sync all file systems.

.. function:: unique\_id()

Returns a string of 12 bytes (96 bits), which is the unique ID of the MCU.

.. function:: usb\_mode([modestr], port=-1, vid=0xf055, pid=-1, msc=(), hid=pyb.hid\_mouse, high\_speed=False)

If called with no arguments, return the current USB mode as a string.

If called with \*modestr\* provided, attempts to configure the USB mode.

The following values of \*modestr\* are understood:

- ``None``: disables USB

- ``'VCP'``: enable with VCP (Virtual COM Port) interface

- ``'MSC'``: enable with MSC (mass storage device class) interface

- ``'VCP+MSC'``: enable with VCP and MSC

- ``'VCP+HID'``: enable with VCP and HID (human interface device)

- ``'VCP+MSC+HID'``: enabled with VCP, MSC and HID (only available on PYBD boards)

For backwards compatibility, ``'CDC'`` is understood to mean

``'VCP'`` (and similarly for ``'CDC+MSC'`` and ``'CDC+HID'``).

The \*port\* parameter should be an integer (0, 1, ...) and selects which

USB port to use if the board supports multiple ports. A value of -1 uses

the default or automatically selected port.

The \*vid\* and \*pid\* parameters allow you to specify the VID (vendor id)

and PID (product id). A \*pid\* value of -1 will select a PID based on the

value of \*modestr\*.

If enabling MSC mode, the \*msc\* parameter can be used to specify a list

of SCSI LUNs to expose on the mass storage interface. For example

``msc=(pyb.Flash(), pyb.SDCard())``.

If enabling HID mode, you may also specify the HID details by

passing the \*hid\* keyword parameter. It takes a tuple of

(subclass, protocol, max packet length, polling interval, report

descriptor). By default it will set appropriate values for a USB

mouse. There is also a ``pyb.hid\_keyboard`` constant, which is an

appropriate tuple for a USB keyboard.

The \*high\_speed\* parameter, when set to ``True``, enables USB HS mode if

it is supported by the hardware.

Constants

---------

.. data:: pyb.hid\_mouse

pyb.hid\_keyboard

A tuple of (subclass, protocol, max packet length, polling interval, report

descriptor) to set appropriate values for a USB mouse or keyboard.

Classes

-------

.. toctree::

:maxdepth: 1

pyb.ADC.rst

pyb.CAN.rst

pyb.DAC.rst

pyb.ExtInt.rst

pyb.Flash.rst

pyb.I2C.rst

pyb.LED.rst

pyb.Pin.rst

pyb.RTC.rst

pyb.Servo.rst

pyb.SPI.rst

pyb.Timer.rst

pyb.UART.rst

pyb.USB\_HID.rst

pyb.USB\_VCP.rst

#tf库

:mod:`tf` --- Tensor Flow

=========================

.. module:: tf

:synopsis: Tensor Flow

The ``tf`` module is capable of executing Quantized TensorFlow Lite Models

on the OpenMV Cam (not supported on the OpenMV Cam M4).

You can read more about how to create your own models that can run on the

OpenMV Cam `here <https://www.tensorflow.org/lite/microcontrollers>`\_\_. In

particular:

\* Supported operations are listed `here <https://github.com/openmv/tensorflow-lib/blob/master/libtf.cc#L71>`\_\_.

\* Note that tensorflow lite operations are versioned. If no version numbers

are listed after the operation then the min and max version supported are

1. If there are numbers after an operation those numbers represent the

minimum and maximum operation version supported.

\* If you are using Keras to generate your model be careful about only using

operators that are supported by tensorflow lite for microcontrollers. Otherwise,

your model will not be runnable by your OpenMV Cam.

\* Convert your model to a FlatBuffer by following the instructions `here <https://www.tensorflow.org/lite/microcontrollers/build\_convert#model\_conversion>`\_\_.

\* Finally, quantize your model by following the instructions `here <https://www.tensorflow.org/lite/microcontrollers/build\_convert#quantization>`\_\_.

Alternatively, just follow Google's in-depth guide `here <https://github.com/openmv/tensorflow/blob/master/tensorflow/lite/micro/examples/person\_detection/training\_a\_model.md>`\_\_.

If you have problems with Google's in-depth guide please contact Google for help.

The final output ``.tflite`` model can be directly loaded and run by your

OpenMV Cam. That said, the model and the model's required sratch RAM must

fit within the available frame buffer stack RAM on your OpenMV Cam.

\* The OpenMV Cam M7 has about 384KB of frame buffer RAM. Please try

to keep your model and it's required scratch buffer under 320 KB.

\* The OpenMV Cam H7 has about 496KB of frame buffer RAM. Please try

to keep your model and it's required scratch buffer under 400 KB.

\* The OpenMV Cam H7 Plus has about 31MB of frame buffer RAM. That

said, running a model anywhere near the that size will be extremely slow.

Alternatively, you can also load a model onto the MicroPython Heap or the OpenMV Cam frame buffer.

However, this significantly limits the model size on all OpenMV Cams.

Functions

---------

.. function:: classify(path, img, [roi, [min\_scale=1.0, [scale\_mul=0.5, [x\_overlap=0, [y\_overlap=0]]]]])

Executes the TensorFlow Lite image classification model on the ``img``

object and returns a list of `tf\_classification` objects. This method

executes the network multiple times on the image in a controllable sliding

window type manner (by default the algorithm only executes the network once

on the whole image frame).

``path`` a path to a ``.tflite`` model to execute on your OpenMV Cam's

disk. The model is loaded into memory, executed, and released all in

one function call to save from having to load the model in the

MicroPython heap. Pass ``"person\_detection"`` to load the built-in

person detection model from your OpenMV Cam's internal flash.

``roi`` is the region-of-interest rectangle tuple (x, y, w, h). If not

specified, it is equal to the image rectangle. Only pixels within the

``roi`` are operated on.

``min\_scale`` controls how much scaling is applied to the network. At the

default value the network is not scaled. However, a value of 0.5 would allow

for detecting objects 50% in size of the image roi size...

``scale\_mul`` controls how many different scales are tested out. The sliding

window method works by multiplying a default scale of 1 by ``scale\_mul``

while the result is over ``min\_scale``. The default value of ``scale\_mul``,

0.5, tests out a 50% size reduction per scale change. However, a value of

0.95 would only be a 5% size reductioin.

``x\_overlap`` controls the percentage of overlap with the next detector

area of the sliding window. A value of zero means no overlap. A value of

0.95 would mean 95% overlap.

``y\_overlap`` controls the percentage of overlap with the next detector

area of the sliding window. A value of zero means no overlap. A value of

0.95 would mean 95% overlap.

.. function:: segment(path, img, [roi])

Executes the TensorFlow Lite image segmentation model on the ``img``

object and returns a list of grayscale `image` objects for each

segmentation class output channel.

``path`` a path to a ``.tflite`` model to execute on your OpenMV Cam's

disk. The model is loaded into memory, executed, and released all in

one function call to save from having to load the model in the

MicroPython heap.

``roi`` is the region-of-interest rectangle tuple (x, y, w, h). If not

specified, it is equal to the image rectangle. Only pixels within the

``roi`` are operated on.

.. function:: detect(path, img, [roi, [thresholds, [invert]]])

Executes the TensorFlow Lite image segmentation model on the ``img``

object and returns a list of `image.blob` objects for each segmentation

class output. E.g. if you have an image that's segmented into two classes

this method will return a list of two lists of blobs that match the requested

thresholds.

``path`` a path to a ``.tflite`` model to execute on your OpenMV Cam's

disk. The model is loaded into memory, executed, and released all in

one function call to save from having to load the model in the

MicroPython heap.

``roi`` is the region-of-interest rectangle tuple (x, y, w, h). If not

specified, it is equal to the image rectangle. Only pixels within the

``roi`` are operated on.

``thresholds`` must be a list of tuples

``[(lo, hi), (lo, hi), ..., (lo, hi)]`` defining the ranges of color you

want to track. You may pass up to 32 threshold tuples in one call. Each tuple

needs to contain two values - a min grayscale value and a max grayscale value.

Only pixel regions that fall between these thresholds will be considered.

For easy usage this function will automatically fix swapped min and max values.

If the tuple is too short the rest of the thresholds are assumed to be at maximum

range. If no thresholds are specified they are assumed to be (128, 255) which

will detect "active" pixel regions in the segmented images.

``invert`` inverts the thresholding operation such that instead of matching

pixels inside of some known color bounds pixels are matched that are outside

of the known color bounds.

.. function:: regression(path, array)

Executes the TensorFlow Lite regression model on the passed array of floats and returns

a new array of floats as the result. This method accepts 1D/2D/3D arrays which must match

the input shape of the network. Arrays should be organized in [height][width][channel] order.

``path`` a path to a ``.tflite`` model to execute on your OpenMV Cam's

disk. The model is loaded into memory, executed, and released all in

one function call to save from having to load the model in the

MicroPython heap.

.. function:: load(path, [load\_to\_fb=False])

``path`` a path to a ``.tflite`` model to load into memory on the MicroPython heap by default.

NOTE! The MicroPython heap is only ~50 KB on the OpenMV Cam M7 and ~256 KB on the OpenMV Cam H7.

Pass ``"person\_detection"`` to load the built-in person detection model from your

OpenMV Cam's internal flash. This built-in model does not use any Micropython Heap

as all the weights are stored in flash which is accessible in the same way as RAM.

``load\_to\_fb`` if passed as True will instead reserve part of the OpenMV Cam frame buffer

stack for storing the TensorFlow Lite model. You will get the most efficent execution

performance for large models that do not fit on the heap by loading them into frame buffer

memory once from disk and then repeatedly executing the model. That said, the frame buffer

space used will not be available anymore for other algorithms.

Returns a `tf\_model` object which can operate on an image.

.. function:: free\_from\_fb()

Deallocates a previously allocated `tf\_model` object created with ``load\_to\_fb`` set to True.

Note that deallocations happen in the reverse order of allocation.

class tf\_classification -- tf classification dection result

-----------------------------------------------------------

The tf\_classification object is returned by `tf.classify()` or `tf\_model.classify()`.

Constructors

~~~~~~~~~~~~

.. class:: tf\_classification()

Please call `tf.classify()` or `tf\_model.classify()` to create this object.

Methods

~~~~~~~

.. method:: rect()

Returns a rectangle tuple (x, y, w, h) for use with `image` methods

like `Image.draw\_rectangle()` of the tf\_classification's bounding box.

.. method:: x()

Returns the tf\_classification's bounding box x coordinate (int).

You may also get this value doing ``[0]`` on the object.

.. method:: y()

Returns the tf\_classification's bounding box y coordinate (int).

You may also get this value doing ``[1]`` on the object.

.. method:: w()

Returns the tf\_classification's bounding box w coordinate (int).

You may also get this value doing ``[2]`` on the object.

.. method:: h()

Returns the tf\_classification's bounding box h coordinate (int).

You may also get this value doing ``[3]`` on the object.

.. method:: classification\_output()

Returns a list of the classification label scores. The size of this

list is determined by your model output channel size. For example,

mobilenet outputs a list of 1000 classification scores for all 1000

classes understood by mobilenet. Use ``zip`` in python to combine

the classification score results with classification labels.

You may also get this value doing ``[4]`` on the object.

class tf\_model -- TensorFlow Model

----------------------------------

If your model size is small enough and you have enough heap or frame buffer space you may wish

to directly load the model into memory to save from having to load it from disk

each time you wish to execute it.

Constructors

.. class:: tf\_model(

Please call `tf.load()` to create the TensorFlow Model object. TensorFlow Model objects allow

you to execute a model from RAM versus having to load it from disk repeatedly.

Methods

.. method:: len()

Returns the size in bytes of the model.

.. method:: ram()

Returns the model's required free RAM in bytes.

.. method:: input\_height()

Returns the input height of the model. You can use this to size your input

image height appropriately.

.. method:: input\_width()

Returns the input width of the model. You can use this to size your input

image width appropriately.

.. method:: input\_channels()

Returns the number of input color channels in the model.

.. method:: input\_datatype()

Returns the model's input datatype (this is a string of "uint8", "int8", or "float").

.. method:: input\_scale()

Returns the input scale for the model.

.. method:: input\_zero\_point()

Returns the output zero point for the model.

.. method:: output\_height()

Returns the output height of the model. You can use this to size your output

image height appropriately.

.. method:: output\_width()

Returns the output width of the model. You can use this to size your output

image width appropriately.

.. method:: output\_channels()

Returns the number of output color channels in the model.

.. method:: output\_datatype()

Returns the model's output datatype (this is a string of "uint8", "int8", or "float").

.. method:: output\_scale()

Returns the output scale for the model.

.. method:: output\_zero\_point()

Returns the output zero point for the model.

.. method:: classify(img, [roi, [min\_scale=1.0, [scale\_mul=0.5, [x\_overlap=0, [y\_overlap=0]]]]])

Executes the TensorFlow Lite image classification model on the ``img``

object and returns a list of `tf\_classification` objects. This method

executes the network multiple times on the image in a controllable sliding

window type manner (by default the algorithm only executes the network once

on the whole image frame).

``roi`` is the region-of-interest rectangle tuple (x, y, w, h). If not

specified, it is equal to the image rectangle. Only pixels within the

``roi`` are operated on.

``min\_scale`` controls how much scaling is applied to the network. At the

default value the network is not scaled. However, a value of 0.5 would allow

for detecting objects 50% in size of the image roi size...

``scale\_mul`` controls how many different scales are tested out. The sliding

window method works by multiplying a default scale of 1 by ``scale\_mul``

while the result is over ``min\_scale``. The default value of ``scale\_mul``,

0.5, tests out a 50% size reduction per scale change. However, a value of

0.95 would only be a 5% size reductioin.

``x\_overlap`` controls the percentage of overlap with the next detector

area of the sliding window. A value of zero means no overlap. A value of

0.95 would mean 95% overlap.

``y\_overlap`` controls the percentage of overlap with the next detector

area of the sliding window. A value of zero means no overlap. A value of

0.95 would mean 95% overlap.

.. method:: segment(img, [roi])

Executes the TensorFlow Lite image segmentation model on the ``img``

object and returns a list of grayscale `image` objects for each

segmentation class output channel.

``roi`` is the region-of-interest rectangle tuple (x, y, w, h). If not

specified, it is equal to the image rectangle. Only pixels within the

``roi`` are operated on.

.. method:: detect(img, [roi, [thresholds, [invert]]])

Executes the TensorFlow Lite image segmentation model on the ``img``

object and returns a list of `image.blob` objects for each segmentation

class output. E.g. if you have an image that's segmented into two classes

this method will return a list of two lists of blobs that match the requested

thresholds.

``roi`` is the region-of-interest rectangle tuple (x, y, w, h). If not

specified, it is equal to the image rectangle. Only pixels within the

``roi`` are operated on.

``thresholds`` must be a list of tuples

``[(lo, hi), (lo, hi), ..., (lo, hi)]`` defining the ranges of color you

want to track. You may pass up to 32 threshold tuples in one call. Each tuple

needs to contain two values - a min grayscale value and a max grayscale value.

Only pixel regions that fall between these thresholds will be considered.

For easy usage this function will automatically fix swapped min and max values.

If the tuple is too short the rest of the thresholds are assumed to be at maximum

range. If no thresholds are specified they are assumed to be (128, 255) which

will detect "active" pixel regions in the segmented images.

``invert`` inverts the thresholding operation such that instead of matching

pixels inside of some known color bounds pixels are matched that are outside

of the known color bounds.

.. method:: regression(array)

Executes the TensorFlow Lite regression model on the passed array of floats and returns

a new array of floats as the result. This method accepts 1D/2D/3D arrays which must match

the input shape of the network. Arrays should be organized in [height][width][channel] order.

#模型

#主代码c++

#include <stdio.h>

#include <stdlib.h>

#include "edge-impulse-sdk/tensorflow/lite/c/builtin\_op\_data.h"

#include "edge-impulse-sdk/tensorflow/lite/c/common.h"

#include "edge-impulse-sdk/tensorflow/lite/micro/micro\_mutable\_op\_resolver.h"

#include "edge-impulse-sdk/porting/ei\_classifier\_porting.h"

#if EI\_CLASSIFIER\_PRINT\_STATE

#if defined(\_\_cplusplus) && EI\_C\_LINKAGE == 1

extern "C" {

extern void ei\_printf(const char \*format, ...);

}

#else

extern void ei\_printf(const char \*format, ...);

#endif

#endif

#if defined \_\_GNUC\_\_

#define ALIGN(X) \_\_attribute\_\_((aligned(X)))

#elif defined \_MSC\_VER

#define ALIGN(X) \_\_declspec(align(X))

#elif defined \_\_TASKING\_\_

#define ALIGN(X) \_\_align(X)

#elif defined \_\_ICCARM\_\_

#define ALIGN(x) \_\_attribute\_\_((aligned(x)))

#endif

#ifndef EI\_MAX\_SCRATCH\_BUFFER\_COUNT

#ifndef CONFIG\_IDF\_TARGET\_ESP32S3

#define EI\_MAX\_SCRATCH\_BUFFER\_COUNT 14

#else

#define EI\_MAX\_SCRATCH\_BUFFER\_COUNT 28

#endif // CONFIG\_IDF\_TARGET\_ESP32S3

#endif // EI\_MAX\_SCRATCH\_BUFFER\_COUNT

#ifndef EI\_MAX\_OVERFLOW\_BUFFER\_COUNT

#define EI\_MAX\_OVERFLOW\_BUFFER\_COUNT 10

#endif // EI\_MAX\_OVERFLOW\_BUFFER\_COUNT

using namespace tflite;

using namespace tflite::ops;

using namespace tflite::ops::micro;

namespace {

#if defined(EI\_CLASSIFIER\_ALLOCATION\_STATIC\_HIMAX) || defined(EI\_CLASSIFIER\_ALLOCATION\_STATIC\_HIMAX\_GNU)

constexpr int kTensorArenaSize = 242640;

#else

constexpr int kTensorArenaSize = 241616;

#endif

#if defined(EI\_CLASSIFIER\_ALLOCATION\_STATIC)

uint8\_t tensor\_arena[kTensorArenaSize] ALIGN(16);

#elif defined(EI\_CLASSIFIER\_ALLOCATION\_STATIC\_HIMAX)

#pragma Bss(".tensor\_arena")

uint8\_t tensor\_arena[kTensorArenaSize] ALIGN(16);

#pragma Bss()

#elif defined(EI\_CLASSIFIER\_ALLOCATION\_STATIC\_HIMAX\_GNU)

uint8\_t tensor\_arena[kTensorArenaSize] ALIGN(16) \_\_attribute\_\_((section(".tensor\_arena")));

#else

#define EI\_CLASSIFIER\_ALLOCATION\_HEAP 1

uint8\_t\* tensor\_arena = NULL;

#endif

static uint8\_t\* tensor\_boundary;

static uint8\_t\* current\_location;

template <int SZ, class T> struct TfArray {

int sz; T elem[SZ];

};

enum used\_operators\_e {

OP\_CONV\_2D, OP\_DEPTHWISE\_CONV\_2D, OP\_PAD, OP\_ADD, OP\_SOFTMAX, OP\_LAST

};

struct TensorInfo\_t { // subset of TfLiteTensor used for initialization from constant memory

TfLiteAllocationType allocation\_type;

TfLiteType type;

void\* data;

TfLiteIntArray\* dims;

size\_t bytes;

TfLiteQuantization quantization;

};

typedef struct {

TfLiteTensor tensor;

int16\_t index;

} TfLiteTensorWithIndex;

typedef struct {

TfLiteEvalTensor tensor;

int16\_t index;

} TfLiteEvalTensorWithIndex;

TfLiteContext ctx{};

static const int MAX\_TFL\_TENSOR\_COUNT = 4;

static TfLiteTensorWithIndex tflTensors[MAX\_TFL\_TENSOR\_COUNT];

static const int MAX\_TFL\_EVAL\_COUNT = 4;

static TfLiteEvalTensorWithIndex tflEvalTensors[MAX\_TFL\_EVAL\_COUNT];

TfLiteRegistration registrations[OP\_LAST];

namespace g0 {

const TfArray<4, int> tensor\_dimension0 = { 4, { 1,96,96,3 } };

const TfArray<1, float> quant0\_scale = { 1, { 0.0039215688593685627, } };

const TfArray<1, int> quant0\_zero = { 1, { -128 } };

const TfLiteAffineQuantization quant0 = { (TfLiteFloatArray\*)&quant0\_scale, (TfLiteIntArray\*)&quant0\_zero, 0 };

const ALIGN(16) int32\_t tensor\_data1[4\*2] = {

0, 0,

0, 1,

0, 1,

0, 0,

};

const TfArray<2, int> tensor\_dimension1 = { 2, { 4,2 } };

const ALIGN(8) int32\_t tensor\_data2[2] = { 32091, -35633, };

const TfArray<1, int> tensor\_dimension2 = { 1, { 2 } };

const TfArray<2, float> quant2\_scale = { 2, { 0.00015335553325712681, 0.00013845304783899337, } };

const TfArray<2, int> quant2\_zero = { 2, { 0,0 } };

const TfLiteAffineQuantization quant2 = { (TfLiteFloatArray\*)&quant2\_scale, (TfLiteIntArray\*)&quant2\_zero, 0 };

const ALIGN(16) int8\_t tensor\_data3[2\*1\*1\*32] = {

/\* [0][0][][] \*/ 56,-44,-45,109,-48,2,-45,76,127,-14,-44,-27,-22,92,-49,33,108,-69,-53,-78,96,68,84,-68,-68,-99,-50,91,10,-75,-18,62,

/\* [1][0][][] \*/ 67,85,81,-73,-103,86,-10,16,-45,-25,42,117,127,42,42,0,-45,0,-73,-103,-87,37,33,121,92,85,-53,-10,112,-119,-119,-19,

};

const TfArray<4, int> tensor\_dimension3 = { 4, { 2,1,1,32 } };

const TfArray<2, float> quant3\_scale = { 2, { 0.0035334315616637468, 0.0031900664325803518, } };

const TfArray<2, int> quant3\_zero = { 2, { 0,0 } };

const TfLiteAffineQuantization quant3 = { (TfLiteFloatArray\*)&quant3\_scale, (TfLiteIntArray\*)&quant3\_zero, 0 };

const ALIGN(16) int32\_t tensor\_data4[32] = { 291, 206, 188, -134, 553, 218, -344, 194, -128, 571, 194, 51, 219, 89, 115, -372, -257, -403, 341, -192, -217, 94, -52, 154, 35, 141, -254, 248, -472, -155, -133, -329, };

const TfArray<1, int> tensor\_dimension4 = { 1, { 32 } };

const TfArray<32, float> quant4\_scale = { 32, { 4.4338128645904362e-05, 4.3867854401469231e-05, 4.1394960135221481e-05, 4.1782121115829796e-05, 4.3021478631999344e-05, 4.0704988350626081e-05, 4.1926010453607887e-05, 4.1938237700378522e-05, 4.1807888919720426e-05, 4.4861881178803742e-05, 4.3881700548809022e-05, 4.2098708945559338e-05, 4.2405066778883338e-05, 4.3749718315666541e-05, 4.1547576984157786e-05, 4.2813306208699942e-05, 4.0334893128601834e-05, 4.2660718463594094e-05, 4.8129168135346845e-05, 4.462332435650751e-05, 4.4292664824752137e-05, 4.0652495954418555e-05, 4.0840419387677684e-05, 4.1789877286646515e-05, 4.2274026782251894e-05, 4.1726758354343474e-05, 4.2773117456817999e-05, 4.5041211706120521e-05, 4.2897972889477387e-05, 3.969723911723122e-05, 4.8145193431992084e-05, 4.2753566958708689e-05, } };

const TfArray<32, int> quant4\_zero = { 32, { 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 } };

const TfLiteAffineQuantization quant4 = { (TfLiteFloatArray\*)&quant4\_scale, (TfLiteIntArray\*)&quant4\_zero, 0 };

const ALIGN(16) int8\_t tensor\_data5[32\*1\*1\*96] = {

/\* [0][0][][] \*/ -58,-66,96,-8,86,0,-76,-110,43,-93,-95,1,50,13,-27,51,-46,-28,-70,-42,-60,-96,7,116,-121,23,83,-52,111,13,-22,-75,75,-65,-79,42,-103,-70,-91,28,82,-82,-111,105,40,-109,111,-22,16,-50,-34,113,-75,-71,-84,111,21,57,111,-101,-77,77,-36,37,-3,30,72,77,60,74,86,-112,94,-97,37,78,-88,16,-53,13,60,9,100,-5,-51,-28,-86,121,36,122,-13,-127,24,79,40,0,

/\* [1][0][][] \*/ -77,28,-68,-18,-29,89,42,11,70,-92,-97,109,70,-21,29,69,113,-37,43,127,-40,-2,40,33,39,29,-35,108,17,15,100,-56,60,104,-3,28,57,65,-6,-64,-56,11,-25,-88,-34,42,-27,-49,-20,-76,121,68,17,80,-69,-40,34,96,65,86,-51,118,-55,-76,-46,43,78,30,112,25,23,15,-107,101,-43,49,92,-41,-36,57,-72,89,-43,-108,33,120,-68,-54,84,86,96,33,91,98,38,-7,

/\* [2][0][][] \*/ 26,33,-62,81,-12,-51,96,100,-114,98,-32,124,100,-11,90,-112,-54,-25,44,69,-94,-13,119,28,-22,95,113,11,-65,95,84,106,-70,-20,-90,-112,-84,103,117,-31,-116,-76,10,1,-88,127,-24,-77,106,48,102,-79,-43,64,32,28,-65,95,-61,9,-12,65,-25,74,107,56,53,78,-88,66,14,-70,39,-67,38,30,-45,-22,9,-5,45,-94,-2,-37,6,-96,-22,-22,100,124,-23,15,-37,-77,-36,-73,

/\* [3][0][][] \*/ 87,6,40,-21,-77,49,-118,43,62,20,66,8,93,97,75,7,107,97,-60,9,116,-91,-67,-119,-17,-118,-87,109,44,-107,89,3,-62,-40,-20,-11,-95,-27,43,49,-28,-98,-97,0,75,27,-49,-88,67,-90,41,-5,96,27,89,-105,-124,-121,-123,-62,-104,36,-41,71,54,-38,-64,-30,-39,-70,4,100,18,-112,57,-51,28,101,97,-53,-96,-127,35,92,29,48,-42,-127,-89,43,-2,-118,93,-93,-98,42,

/\* [4][0][][] \*/ 74,118,19,77,-20,-28,44,46,84,103,5,68,35,-23,2,-49,-13,8,-36,1,113,-74,101,38,0,-7,74,-34,-41,11,-52,-48,81,127,30,-84,-37,3,-77,63,-39,-46,-49,-79,50,114,46,77,35,-51,-18,-85,-62,60,-81,24,62,-77,59,27,14,-52,-46,-55,-42,60,18,-18,19,-72,-74,-68,-39,31,117,72,48,-6,7,-32,15,-33,-99,-13,13,77,106,-5,-111,-68,67,46,29,96,-105,12,

/\* [5][0][][] \*/ 80,-103,83,71,50,-111,33,61,100,-114,3,-68,30,108,42,42,70,66,-70,114,39,57,22,-23,-107,-56,-13,-94,57,42,18,-96,-41,3,-67,17,-115,-33,56,22,-89,-51,39,108,-37,30,38,-51,-111,118,34,43,47,15,45,-15,119,48,-54,-38,39,-68,49,-75,-23,114,-25,18,87,-2,114,-70,-47,91,-25,-74,20,-47,16,67,118,48,37,120,-81,91,-8,50,78,-98,127,103,-11,35,87,-90,

/\* [6][0][][] \*/ 35,-45,112,84,-91,-71,-7,64,-10,-89,1,-93,84,53,113,-125,37,-2,-4,98,-84,-126,107,55,-67,-91,-14,-61,-32,-122,44,-83,-95,-127,-51,-109,-14,34,-115,79,82,54,24,-21,-118,103,-98,-50,-90,13,-30,-73,57,-56,91,-34,4,-7,-71,111,-102,-1,62,-74,-57,62,99,-47,75,-3,-112,-8,53,-76,-96,-45,-88,-108,-62,-21,29,-31,39,-32,-78,-95,-43,-103,81,90,95,14,-58,-74,71,-24,

/\* [7][0][][] \*/ 106,66,-36,-52,-69,127,-15,2,114,12,-94,17,61,-87,52,79,59,20,3,124,61,40,-2,94,-69,114,47,60,-76,117,-17,45,29,29,74,101,-103,96,-5,-27,103,-76,85,77,53,3,109,-112,59,-53,-106,-60,-22,87,25,37,125,-92,4,-55,12,-4,90,-115,-64,-50,73,20,-107,17,73,90,92,32,65,64,-58,60,-98,38,99,-35,-11,-69,94,41,-113,105,89,14,90,75,-52,-34,6,-72,

/\* [8][0][][] \*/ 35,40,-45,-73,51,18,-81,-24,20,31,42,94,-4,-50,-90,-77,-76,95,-74,8,26,-27,81,108,-123,87,43,76,-76,10,-93,111,96,-33,43,-3,114,-1,-27,75,79,101,-110,-127,-7,-34,-26,108,20,-104,-72,-13,-71,-9,-40,-116,42,112,-103,103,109,40,33,110,-44,-55,-24,117,-97,56,28,-7,74,64,5,22,-58,106,82,94,31,79,-122,-94,22,39,78,-53,32,-45,-89,95,-90,-110,-57,67,

/\* [9][0][][] \*/ 33,8,125,64,92,-89,-29,-103,57,52,1,85,7,-63,-80,-18,22,12,-81,-54,29,111,-92,-81,108,-9,13,46,4,-50,-72,106,68,6,43,-24,-64,26,-24,-28,56,-52,59,-82,87,20,-91,65,59,8,-62,-29,87,-66,-60,110,110,-56,-6,6,-76,114,91,-67,-91,12,117,58,-34,104,127,-65,7,81,-85,-77,-81,-103,-22,110,48,27,125,28,-29,-54,41,-15,120,38,-15,122,118,25,58,-31,

/\* [10][0][][] \*/ -81,75,91,-82,-6,113,111,29,27,39,28,-62,104,86,-29,15,-28,82,-89,-49,-12,31,60,-62,36,65,-110,-109,-41,89,-91,-71,84,99,123,71,-34,12,-25,33,-30,118,-113,-30,-97,-52,0,-55,0,11,45,22,67,20,-54,-24,-93,-18,35,-40,3,83,120,42,127,-66,-92,-33,32,-23,-87,-61,89,75,-77,107,-4,67,69,6,82,5,63,108,-94,73,98,-14,42,16,-1,-57,-43,-39,-32,-68,

/\* [11][0][][] \*/ 87,1,-15,85,127,55,56,31,99,101,-79,116,-5,-85,30,-65,-55,-108,72,50,34,76,25,-109,-106,-90,-20,74,85,1,25,-9,-93,-78,56,40,-61,79,-23,-65,-25,107,-83,9,-42,-59,-98,-89,77,113,120,60,-91,-102,25,31,-63,-63,-28,-88,-77,92,94,8,38,94,-58,13,97,54,-84,-71,42,113,-98,52,-60,35,-45,74,-104,-25,-66,-40,49,-80,8,-17,-92,-3,15,-96,-82,-26,-66,-44,

/\* [12][0][][] \*/ -7,64,-53,-15,98,18,-30,38,-87,69,-9,20,-39,18,54,-75,-94,-29,-41,-70,34,-6,22,105,-63,-38,107,-97,101,25,-34,126,-86,-69,-39,16,4,-70,-10,44,123,45,-114,-33,-21,-78,-37,123,-42,-108,-69,117,44,0,-74,96,-13,-61,59,73,66,120,6,57,124,43,-42,109,67,101,48,79,90,126,-11,-2,-26,40,-113,-110,-52,110,56,22,-31,39,55,127,-87,-102,98,53,112,-6,41,75,

/\* [13][0][][] \*/ 90,-65,100,84,96,125,82,61,63,-94,15,-29,92,72,69,-47,-112,-69,-68,39,71,-26,117,-41,-67,50,9,16,-84,-76,80,45,39,102,-15,32,-80,-23,29,65,-58,76,-60,17,-43,31,1,-31,35,103,-28,-90,-97,26,108,-72,-44,21,-27,-72,-23,113,-64,9,-43,-55,-90,77,-33,-21,96,-2,-89,45,-76,-42,-93,67,81,-73,54,-95,103,25,-78,-94,88,127,-6,-44,-77,62,117,-41,78,-15,

/\* [14][0][][] \*/ -86,-104,94,58,87,-23,-14,-117,-104,-31,74,89,-27,-85,-90,-89,-50,-66,18,33,-40,36,1,67,-124,-68,-33,34,-113,-115,-80,44,107,-24,-32,0,104,124,40,-124,-4,-114,24,98,73,-116,65,-85,122,47,83,62,104,-17,-10,18,-48,-1,50,102,-14,-80,-21,61,-90,-9,82,-123,-88,-107,-88,101,3,90,58,-75,64,-108,58,29,50,-22,-116,-41,-117,-90,-127,114,118,10,96,-51,93,110,-15,48,

/\* [15][0][][] \*/ 38,-28,-33,89,86,-94,55,-102,69,-8,-77,4,101,-4,-97,-117,83,5,-99,17,42,13,67,102,78,61,33,88,15,50,61,-107,2,5,-69,-8,-74,107,104,-71,10,106,-51,-30,23,-83,-49,-86,-12,-87,-9,23,-105,-103,-57,-15,62,53,-53,104,-50,-72,-7,100,-42,91,-77,13,-67,4,57,-80,13,-20,53,-112,-9,50,100,106,-88,54,-115,-68,-127,-43,-96,31,79,65,-118,-46,48,105,-113,114,

/\* [16][0][][] \*/ -48,8,108,-44,34,110,-63,40,65,43,-119,69,-21,-57,-60,-69,108,81,67,57,111,81,-46,50,-43,46,-78,64,-115,3,41,5,-112,82,54,63,-43,52,-61,-72,18,-54,80,90,68,104,107,-18,-63,-127,-2,106,-50,125,63,39,22,98,65,9,117,110,25,64,46,6,-42,114,35,107,-39,-16,28,86,61,91,-23,101,-120,6,8,-94,54,6,7,101,109,0,76,7,-20,111,65,-42,-100,30,

/\* [17][0][][] \*/ 11,-117,27,60,-44,-117,-76,-106,23,-90,74,52,101,57,-120,32,27,45,104,115,-97,-23,5,-58,1,65,-85,26,107,-55,-43,73,-38,-37,21,2,-54,92,-90,-46,-107,-46,-33,-16,2,5,69,-119,78,108,-18,23,-21,67,98,-77,-124,63,-86,-127,29,-94,72,96,-91,-81,33,-122,-111,-20,96,27,5,-123,-70,59,116,-58,33,-81,5,-23,-80,-101,-122,65,65,-39,18,68,-100,-122,-15,93,5,-1,

/\* [18][0][][] \*/ -85,91,111,-34,12,-66,-78,31,17,69,-92,-6,-21,38,64,-43,35,-18,64,-20,92,-26,30,-59,38,-79,-8,126,52,29,5,34,41,-26,-64,-95,-16,30,-9,65,-35,6,-27,-4,-101,13,21,13,18,-3,61,27,-77,-11,2,78,-40,33,0,-92,77,127,13,34,-57,-105,-12,103,80,-45,-33,104,-85,-34,-65,-81,47,62,8,-34,-46,-4,-23,-61,97,-26,70,57,-92,-68,-38,89,82,-29,40,3,

/\* [19][0][][] \*/ 49,35,72,51,54,-83,91,88,99,-60,29,-13,-91,31,-54,59,68,-97,104,54,-22,77,-54,11,77,-77,26,-65,-89,-114,-60,44,89,-73,31,0,6,-59,-54,-92,46,-110,-73,8,-74,30,90,-89,-23,34,-37,41,-34,32,5,-9,22,113,-2,-26,53,-54,93,65,-66,-3,-25,15,-9,57,71,-49,-57,-111,-54,-76,-24,-109,14,127,55,-109,-30,75,-83,-8,18,-5,84,29,-4,61,-34,107,69,-18,

/\* [20][0][][] \*/ 13,73,66,107,-89,83,16,55,25,7,-85,0,-11,81,37,-73,-53,-37,-68,-37,19,-45,-76,-71,73,-75,-103,43,17,6,-39,104,97,43,-18,57,36,-32,-20,-103,-83,-36,-84,-102,-24,-65,-63,99,59,-120,-104,32,42,110,45,-4,85,73,89,83,103,-56,-71,-35,-33,-68,-51,27,-87,-73,-69,72,-64,-86,25,17,100,72,103,127,79,1,77,-27,55,109,-13,66,15,89,23,58,17,20,-10,9,

/\* [21][0][][] \*/ 81,74,75,-121,106,-72,79,19,-11,-63,-74,43,9,42,-53,-95,-107,-32,33,-7,117,-52,75,-85,59,-83,-33,77,18,12,-9,-15,7,22,-94,-122,119,104,-118,-37,-5,8,26,-11,-108,-83,-86,98,-25,-68,0,100,41,73,-93,32,-49,53,-56,127,61,-81,20,-15,92,38,-38,-82,-109,-8,66,83,37,-18,75,71,-111,-31,67,97,17,106,42,-72,28,-100,64,-80,-107,88,114,-112,-44,-61,81,-36,

/\* [22][0][][] \*/ 13,-62,-77,-119,91,-53,116,52,-119,88,-80,-118,107,-126,-121,-93,41,85,-117,-65,86,-79,32,35,97,-2,-123,98,-61,-81,-73,-120,49,62,24,10,4,-19,-70,-125,-1,-20,-2,99,55,-127,93,-28,-69,-85,-10,-14,-72,-100,-50,30,-14,-40,-40,-33,100,-82,-78,-121,-69,19,-121,-52,122,106,-27,-13,-76,8,65,-99,52,-55,68,96,111,-109,-87,-125,82,86,101,-36,-64,33,-87,53,17,25,-31,102,

/\* [23][0][][] \*/ 39,-103,3,39,107,-108,8,-109,44,-76,91,81,23,-107,22,-39,95,-34,-103,114,100,-83,108,-31,-13,-4,88,22,78,60,71,-30,-43,37,25,29,104,-4,-109,2,-104,2,3,-108,-78,-90,117,29,79,-105,-90,-42,87,-80,-47,100,-82,-26,106,49,-113,-92,-44,-15,46,46,-119,-127,73,-21,118,-71,-5,-30,-37,-99,57,-21,98,96,-50,-52,31,-104,100,12,114,-10,-87,50,110,3,115,-45,-16,81,

/\* [24][0][][] \*/ -99,-61,-104,-79,-79,-106,-68,-90,61,127,86,49,123,65,-40,-116,-81,42,35,-47,100,-81,-55,-35,57,-33,76,62,113,64,2,-61,2,6,56,23,-51,-96,71,-23,-80,-100,102,8,14,-35,29,76,-87,-41,-21,23,78,51,77,-85,3,73,30,3,95,-3,43,95,-94,103,-46,-100,-99,50,-54,-24,-36,40,99,106,71,-75,-104,-103,-77,-93,-60,-114,4,63,-17,-96,31,-106,10,-75,-15,33,-91,26,

/\* [25][0][][] \*/ -113,-63,-80,-20,106,-67,-103,61,46,-106,-36,25,80,2,-113,-18,-72,65,43,-102,72,89,-9,-57,25,-47,-103,44,85,-82,43,127,99,-108,49,75,105,-65,-15,-15,-85,105,-104,7,102,49,59,42,-107,104,-73,12,52,-19,-57,-2,-14,56,-104,82,127,48,61,52,13,85,-16,-9,-83,0,-22,-112,39,-110,74,26,83,78,19,-1,57,58,-47,92,-73,-51,-106,-85,-19,-35,112,31,-114,-10,-14,-77,

/\* [26][0][][] \*/ 75,-112,-22,2,-103,-11,-14,-15,-97,-127,103,84,-82,-51,7,-17,110,28,-84,-74,10,70,97,10,51,56,10,46,9,-1,-94,-7,-31,-62,-103,-103,-69,81,-56,111,-103,101,61,28,-71,70,-44,101,-24,25,-10,92,8,-45,-9,-113,93,67,54,23,-1,-97,58,-21,109,-25,-58,69,-93,84,85,94,78,-103,-89,7,40,-111,-99,26,-32,73,77,-79,77,-4,-43,7,-87,109,9,13,-61,108,44,93,

/\* [27][0][][] \*/ 44,10,-50,-55,-119,-69,-53,4,78,91,48,-75,-25,56,90,-25,62,-7,54,-87,-79,21,-25,-10,-99,38,-82,-2,54,16,-22,-4,69,76,-72,-84,-90,-48,-103,107,-79,28,31,17,108,25,79,105,96,55,64,-74,-91,-70,-5,-26,106,-22,-97,-73,-51,-64,-39,54,28,-42,-94,127,41,12,-103,16,57,-55,-8,-75,35,-61,97,109,91,-30,113,-46,-63,-56,-29,29,-39,1,-36,-37,47,35,80,55,

/\* [28][0][][] \*/ 82,-14,75,33,-114,-66,-110,-84,-49,-112,47,66,-81,-95,-8,88,83,100,-74,67,-34,11,-120,-86,-127,69,-93,-66,-77,-78,-13,-25,-113,5,70,64,6,-119,-46,19,92,30,-104,38,-119,-21,2,37,11,-64,-80,-41,-90,14,44,31,56,-24,72,40,-13,-50,-98,4,50,62,-34,71,-80,19,-113,-46,57,29,-12,-111,-122,-76,-98,34,-83,-102,105,-39,50,-119,53,-92,-108,-82,-43,-67,-42,48,-105,-75,

/\* [29][0][][] \*/ 80,64,61,-39,-101,9,-127,8,121,-20,-96,-101,113,-112,-53,-49,-109,111,25,-76,-81,-36,58,4,-40,2,-47,-30,-113,-13,114,-7,57,4,-87,-56,-14,110,-13,107,51,115,-44,86,1,112,-54,-27,20,-12,58,-107,75,-103,116,47,-103,64,-56,116,-126,101,-106,-54,-73,-71,57,111,97,-88,36,-78,119,75,63,28,119,-68,-32,53,-7,-91,-11,-83,68,125,51,81,-41,103,76,84,-114,17,-5,68,

/\* [30][0][][] \*/ 50,-58,-57,31,-82,21,-59,-14,126,-15,-2,-36,-85,51,50,39,-7,59,89,-92,66,-91,-103,10,-72,18,-58,102,11,-78,-82,9,-76,-61,-99,-47,10,-14,-5,24,84,-54,14,86,92,-11,108,57,-66,76,36,-71,91,-17,-11,90,-45,-29,-46,-19,-65,-6,-43,-10,-78,-59,-87,83,-19,127,46,-6,-62,93,35,-79,-41,-67,-69,-53,37,52,-61,4,-96,31,-69,122,-4,-80,69,70,-12,4,36,-38,

/\* [31][0][][] \*/ 102,-95,33,110,48,59,-64,-99,-18,-23,-84,-92,105,-127,-52,108,-66,-94,39,47,60,32,-35,103,42,-118,32,37,-77,-76,44,108,20,19,-96,23,93,-34,-69,-115,57,-73,-33,28,3,-57,-43,22,69,22,101,-34,-118,-43,-85,22,-122,72,4,111,-40,-7,83,-76,70,108,-49,-76,70,-120,-124,7,-99,-44,0,2,-63,-51,46,116,-80,-43,54,71,53,93,-99,13,14,-56,-88,-56,8,100,-7,-21,

};

const TfArray<4, int> tensor\_dimension9 = { 4, { 16,1,1,96 } };

const TfArray<16, float> quant9\_scale = { 16, { 0.038387209177017212, 0.02074945904314518, 0.0093489484861493111, 0.010261178016662598, 0.017564577981829643, 0.014997696503996849, 0.017279677093029022, 0.012937434948980808, 0.011909771710634232, 0.012096630409359932, 0.0340854711830616, 0.031734004616737366, 0.017524749040603638, 0.0085630202665925026, 0.018738718703389168, 0.022888790816068649, } };

const TfArray<16, int> quant9\_zero = { 16, { 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 } };

const TfLiteAffineQuantization quant9 = { (TfLiteFloatArray\*)&quant9\_scale, (TfLiteIntArray\*)&quant9\_zero, 0 };

const ALIGN(16) int32\_t tensor\_data10[96] = { 19792, 63372, -2261, -376, 13682, -4981, 1201, -273, 6050, 11103, -3392, 213, 758, 4234, 5195, -9402, -4360, -25, -416, 12755, -11406, 14167, -3092, 7114, 25814, 788, -6133, 19554, 13536, -19188, 1489, 22085, 23517, -12543, 1685, -578, -350, -1030, 4164, -7115, -5732, -932, -8435, 75, 29389, -847, 11994, -774, -14767, 690, -206, 245, -1984, 20516, -3143, 16889, 16770, 269, 35254, -71, -8743, -981, 467, -227, -1286, 15806, -1034, -7328, -1933, 4079, 717, -3639, 1463, 460, -2328, -5443, -590, -2200, 36196, -7554, -6645, -11884, -585, 532, 54981, -12764, -10092, -27611, -7317, 8865, -3632, 33412, -249, -1978, -2563, 1118, };

const TfArray<1, int> tensor\_dimension10 = { 1, { 96 } };

const TfArray<96, float> quant10\_scale = { 96, { 0.00011711697152350098, 8.8770560978446156e-05, 0.00012862926814705133, 0.00020906103600282222, 0.00025019724853336811, 0.00016367147327400744, 4.7838293539825827e-05, 8.5153791587799788e-05, 0.00024014423252083361, 0.00021987481159158051, 0.00029133612406440079, 0.00012078744475729764, 0.00031450900132767856, 0.00021356073557399213, 0.00041074323235079646, 0.00014966502203606069, 0.00020692012913059443, 0.00019081671780440956, 0.00016910342674236745, 9.3474023742601275e-05, 6.8872679548803717e-05, 4.3476553400978446e-05, 0.00014026372809894383, 0.00010448208922753111, 9.505151683697477e-05, 0.00011016939242836088, 0.00015721007366664708, 0.00015454273670911789, 7.6089003414381295e-05, 0.0004864918882958591, 0.00036424788413569331, 0.00025921082124114037, 0.00011840178194688633, 0.00011801140499301255, 0.0002647035289555788, 0.00040113861905410886, 0.00036448868922889233, 0.00018960214219987392, 0.00022158997307997197, 0.00010879476030822843, 0.00020616032998077571, 0.00025537310284562409, 0.00035294998087920249, 7.1762799052521586e-05, 0.00038453174056485295, 0.0001032355212373659, 0.00010313157690688968, 0.00026261768653057516, 0.00072702264878898859, 0.00011119018017780036, 0.00049944798229262233, 0.00020226760534569621, 0.00025771098444238305, 0.0001341525639872998, 0.00026989585603587329, 0.00010970600123982877, 0.00016568588034715503, 0.00044895822065882385, 5.1704500947380438e-05, 0.00015774203347973526, 0.0003109700046479702, 0.00029595699743367732, 9.9102770036552101e-05, 0.00014917069347575307, 0.00016565316764172167, 0.00011071167682530358, 0.00033140092273242772, 0.0002891917247325182, 0.00022841374448034912, 0.00011302627535769716, 0.00027641741326078773, 9.5078568847384304e-05, 0.00028555205790325999, 0.0001332979736616835, 0.00033387838630005717, 0.00033614659332670271, 0.00013369024964049459, 8.2679085608106107e-05, 7.0391412009485066e-05, 0.00016676202358212322, 0.00017736172594595701, 0.00021954718977212906, 0.0001506031840108335, 0.00026080870884470642, 6.4632353314664215e-05, 0.00011997654655715451, 7.6878503023181111e-05, 0.00047493955935351551, 0.00023832148872315884, 0.00012591113045345992, 0.00029827110120095313, 3.2331845432054251e-05, 0.00018272377201355994, 0.00011142082803416997, 0.00037295918446034193, 0.00028826264315284789, } };

const TfArray<96, int> quant10\_zero = { 96, { 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 } };

const TfLiteAffineQuantization quant10 = { (TfLiteFloatArray\*)&quant10\_scale, (TfLiteIntArray\*)&quant10\_zero, 0 };

const ALIGN(16) int8\_t tensor\_data11[1\*3\*3\*96] = {

/\* [0][0][][] \*/ -46,-73,-9,127,22,71,75,-58,-16,-18,-15,-5,-30,2,37,3,-9,-45,26,22,17,-127,127,6,-80,5,50,-29,-127,-127,38,23,12,28,61,-18,-31,-10,32,-7,-14,28,-109,70,25,-70,37,45,-17,53,-23,19,3,-6,15,47,-92,-49,-65,-19,-9,1,11,38,68,-26,76,13,-10,16,20,127,23,22,26,0,-10,18,127,-6,-13,15,-26,19,125,19,-76,-16,-2,-30,-14,-40,2,-25,-16,25, 127,-127,77,106,-43,127,39,-116,70,5,-18,127,127,-96,-13,17,10,127,127,-111,127,-78,81,67,-127,-120,-65,-35,-55,46,-60,-19,-89,53,-127,-84,-44,-14,-52,62,0,127,127,82,-49,-115,-111,-51,-14,-127,43,84,67,-32,44,83,-47,42,-80,-111,-43,-41,-61,66,73,-62,28,30,-81,32,-17,92,118,127,13,20,2,59,-33,53,66,127,127,-24,-8,72,-111,127,-123,-98,5,-127,-89,-127,-39,106, -74,-12,30,-82,15,-25,63,-7,81,-35,-27,18,-20,-3,-8,-11,-9,-16,6,-4,-23,-104,30,-2,-121,96,9,-55,119,50,-15,9,1,15,14,-29,-9,35,21,13,25,12,-115,63,19,-18,37,13,-14,11,-16,5,-10,1,-4,21,-66,6,-36,50,4,41,34,-1,-6,-40,-31,24,9,2,-3,36,10,-26,-5,4,-12,-6,-84,2,-21,-23,-18,2,-111,8,-33,-23,-52,25,-25,-32,29,-27,-14,1,

/\* [0][1][][] \*/ -63,-75,75,-35,-48,93,18,127,-53,-22,-1,-119,26,-113,78,31,-3,-60,-17,89,53,-70,48,50,-93,-14,127,-23,-24,63,122,-105,-127,90,-85,-55,-37,32,-29,96,-49,-25,-7,47,-47,127,-24,64,16,97,20,-2,14,-54,27,31,127,121,-74,-84,127,-32,-35,127,71,-39,-15,-1,127,38,127,97,-41,102,-4,11,14,13,-30,9,-42,123,-79,111,2,34,-55,31,78,-127,-31,-8,127,-70,-23,0, 67,126,58,-113,127,83,89,39,-127,127,127,-38,-66,-127,-127,127,127,-62,-124,-50,106,93,-121,127,-105,-92,-43,127,-127,109,-127,-127,-88,127,106,127,127,-127,-108,127,-64,-96,108,-127,-127,-35,-127,-127,127,-111,86,42,127,127,127,-127,101,-127,-127,127,20,-81,-127,-99,-127,127,-127,127,-125,-127,-107,-83,-127,-81,127,127,-127,127,-77,127,127,30,-42,-127,60,127,94,46,127,11,127,1,-67,36,127,-127, -52,-13,-50,38,-56,-99,34,-27,19,-35,-28,95,1,94,14,10,-4,63,-7,127,2,-87,-40,6,-72,106,-71,-20,-69,-16,17,-61,-70,39,6,-17,-30,46,-69,43,127,-4,-36,29,-40,-60,16,43,8,119,-11,6,-38,-71,3,19,95,2,-1,22,-34,127,92,-51,-42,12,38,-10,-77,120,-18,-87,-1,-55,-16,-21,27,16,75,-37,-71,-41,57,14,-46,-17,71,22,14,62,-43,21,-43,97,-26,-5,

/\* [0][2][][] \*/ -68,2,-3,-21,13,-44,64,25,54,-29,-36,-3,18,20,-26,3,-7,63,7,1,-56,-85,3,-9,-25,-57,35,-68,89,38,25,25,16,-7,-10,-45,-7,36,-4,0,-16,-5,-69,103,4,31,11,35,-15,-31,11,-20,22,-1,8,15,-71,23,-18,13,14,1,-20,-4,-1,-48,-47,14,-32,13,26,29,13,-20,28,4,17,5,-77,-13,-7,-7,18,14,-127,4,13,-43,-60,-48,-26,-51,10,57,-18,-8, 120,34,-127,-36,-72,-24,37,31,-67,-29,-20,-109,-78,114,-13,38,2,-52,-10,-125,93,-50,-38,0,-107,127,-51,20,-37,-22,-9,-4,-62,17,28,5,-23,-27,127,54,11,-14,67,-35,-37,76,-68,-44,-32,-5,-127,-127,-50,-27,1,38,-65,-7,9,38,-8,-19,9,-48,-53,32,65,-23,117,107,-23,-71,11,-59,4,4,99,5,51,21,13,-96,-49,2,4,7,127,40,-63,39,-18,-109,41,80,-10,26, -54,13,-17,25,15,-69,-127,2,-12,-9,-15,5,15,-24,19,0,-8,-18,4,17,-49,-73,14,0,20,-99,20,3,-50,-40,-6,16,42,-19,-6,-25,-36,4,42,-40,23,-2,-46,-34,3,64,22,15,-9,-4,10,-9,17,5,-5,-6,-62,-13,0,-40,-10,10,85,-11,36,-2,-2,28,2,4,-3,19,-18,15,0,15,36,1,76,-12,3,-8,-7,-18,76,-1,42,-39,58,38,-16,-56,-8,3,-14,-16,

};

const TfArray<4, int> tensor\_dimension13 = { 4, { 96,1,1,16 } };

const TfArray<96, float> quant13\_scale = { 96, { 0.00060403492534533143, 0.00049354881048202515, 0.00040563297807238996, 0.00024152023252099752, 0.00052352645434439182, 0.0011885659769177437, 0.00115803605876863, 0.0012649776181206107, 0.00024500690051354468, 0.00084905169205740094, 0.00079547514906153083, 0.00053482892690226436, 0.00050751259550452232, 0.00056039524497464299, 0.00021670655405614525, 0.00098992977291345596, 0.00068433472188189626, 0.00042590626981109381, 0.0005582791636697948, 0.00069243262987583876, 0.00089952489361166954, 0.0017440621741116047, 0.0004737183335237205, 0.0016198338707908988, 0.0017405498074367642, 0.00062949681887403131, 0.00062686012824997306, 0.00070697005139663815, 0.00075768801616504788, 0.00030505453469231725, 0.00015771118341945112, 0.00042545475298538804, 0.00064576294971629977, 0.00100353779271245, 0.00050674600061029196, 0.00086972216377034783, 0.0012341131223365664, 0.0012109170202165842, 0.00065618602093309164, 0.0019233140628784895, 0.00048355301260016859, 0.00056694564409554005, 0.0001691017096163705, 0.00087399675976485014, 0.00044277976849116385, 0.00051924615399912, 0.00091477629030123353, 0.0007587699219584465, 0.0002122443838743493, 0.00046106555964797735, 0.00080838979920372367, 0.00066463800612837076, 0.0011534804943948984, 0.00075168348848819733, 0.0013565070694312453, 0.00067533878609538078, 0.00067954085534438491, 0.00033181646722368896, 0.0010110255097970366, 0.00054142711451277137, 0.0008372893207706511, 0.0002366238331887871, 0.00055961066391319036, 0.00049907679203897715, 0.00075670058140531182, 0.00088512053480371833, 0.00064986658981069922, 0.00065830216044560075, 0.00059649191098287702, 0.00081387232057750225, 0.00065729249035939574, 0.00055097887525334954, 0.00033710451680235565, 0.00062274298397824168, 0.0010419808095321059, 0.00082871789345517755, 0.00074483914067968726, 0.00091787223936989903, 0.0010414643911644816, 0.00097760104108601809, 0.00039568799547851086, 0.00038369419053196907, 0.00067813100758939981, 0.00047729193465784192, 0.0012344871647655964, 0.0008899391395971179, 0.00088268826948478818, 0.00027377755031920969, 0.00047249736962839961, 0.00062891747802495956, 0.00074036099249497056, 0.0009435752872377634, 0.00072618864942342043, 0.00054326211102306843, 0.00082194508286193013, 0.00051448715385049582, } };

const TfArray<96, int> quant13\_zero = { 96, { 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 } };

const TfLiteAffineQuantization quant13 = { (TfLiteFloatArray\*)&quant13\_scale, (TfLiteIntArray\*)&quant13\_zero, 0 };

const ALIGN(16) int32\_t tensor\_data14[16] = { 355, -3181, -9421, 29457, 24989, 875, -5375, -5595, 19657, 5088, 31884, 18105, -11005, 13376, 34498, -26474, };

const TfArray<1, int> tensor\_dimension14 = { 1, { 16 } };

const TfArray<16, float> quant14\_scale = { 16, { 0.00041951253660954535, 0.0002555291575845331, 0.00026252164389006793, 0.00020834882161580026, 0.000344826839864254, 0.00028655384085141122, 0.00033012317726388574, 0.00039030503830872476, 0.00022032063861843199, 0.00028337282128632069, 0.0003609928535297513, 0.00072178716072812676, 0.0003023383324034512, 0.00045209217933006585, 0.000390946603147313, 0.00032538696541450918, } };

const TfArray<16, int> quant14\_zero = { 16, { 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 } };

const TfLiteAffineQuantization quant14 = { (TfLiteFloatArray\*)&quant14\_scale, (TfLiteIntArray\*)&quant14\_zero, 0 };

const ALIGN(16) int8\_t tensor\_data15[16\*1\*1\*96] = {

/\* [0][0][][] \*/ -8,15,-7,23,50,78,-24,-32,45,6,7,-3,-5,5,29,68,4,-9,-10,-10,94,-54,7,-19,-31,-39,-43,-106,-8,-1,42,53,119,1,-2,-6,7,-17,-30,5,-28,14,18,-33,3,-26,-14,30,-15,15,11,-6,-28,-11,0,-8,34,28,-18,-37,-58,25,-22,-3,14,12,57,-2,15,1,-4,-29,-79,-10,-9,3,-13,-95,10,-22,41,-25,-59,-42,28,-13,12,24,-17,9,23,-36,28,21,-8,127,

/\* [1][0][][] \*/ 76,-90,58,76,7,-76,4,-46,-11,-88,-4,15,6,-48,-32,-55,-92,-2,11,33,-22,-117,-32,22,127,-51,-60,16,-26,90,-18,69,20,5,-45,21,-16,-3,62,-15,3,-75,62,-21,-56,21,70,5,36,20,2,-67,68,-64,51,-18,17,20,-17,-107,41,-67,28,-42,12,-86,42,51,48,68,-47,-6,-124,44,-21,78,40,105,23,-20,48,6,-88,37,-89,60,-4,71,-57,9,14,2,13,-60,39,5,

/\* [2][0][][] \*/ 20,14,5,-36,3,11,-15,30,61,-6,19,73,7,1,44,0,-4,-16,14,-6,-24,-69,7,-51,-8,-11,-22,26,-3,30,127,-55,13,-65,-21,51,18,118,-47,-25,-3,-16,-20,34,-40,-54,-57,20,31,30,-36,-21,31,0,-7,-26,-20,3,-8,29,2,-2,66,89,11,-79,14,-69,-20,-16,-10,11,-10,13,-23,9,18,52,9,14,10,84,18,-11,-17,7,19,-3,24,1,-9,-53,-46,-2,-15,61,

/\* [3][0][][] \*/ -12,-23,-47,29,27,-13,-20,-32,-11,19,10,18,17,-19,46,-9,48,-42,23,14,-13,-17,-12,-12,-15,13,4,-7,24,-40,69,-23,10,-23,-14,56,-7,-4,-7,14,7,-68,11,-32,-15,-5,4,-2,-127,41,-22,35,-122,-59,-24,-36,32,9,-9,53,-7,16,20,17,-8,-96,-44,19,51,10,-1,25,-25,17,-2,-47,-41,-58,32,-41,12,-23,12,77,46,65,35,6,39,-9,-11,12,-36,9,-23,-34,

/\* [4][0][][] \*/ 1,-25,8,-79,-4,-5,-50,4,-15,-41,-23,1,8,69,3,12,54,-6,3,3,-2,33,52,5,-10,-1,18,18,3,-2,0,-9,-42,5,3,11,40,-29,2,-16,8,9,-99,-26,34,-11,-10,-13,7,83,-21,-15,-30,-14,0,-6,1,-34,28,41,11,9,11,-11,43,-109,23,-20,-46,5,-10,-1,-2,21,-17,21,-26,74,51,9,5,10,127,-38,42,-22,-12,-14,76,9,-4,19,-39,25,-10,-40,

/\* [5][0][][] \*/ -5,6,10,3,-2,-16,0,120,52,-26,-17,-41,-19,31,33,34,-29,-10,1,-41,-47,-5,1,-47,22,7,27,74,69,-2,66,47,-42,10,-2,6,-56,-56,-23,-17,-32,-19,-3,-3,71,21,7,15,11,11,-16,-3,-36,-18,9,-8,-32,-21,-14,32,-54,28,-14,-48,5,15,-14,8,2,19,6,-3,31,7,7,0,-9,31,4,127,-8,-41,32,29,8,-3,16,-29,-4,-33,20,-124,-35,-20,-2,-117,

/\* [6][0][][] \*/ -23,-8,-12,-58,-9,-13,-23,6,1,20,-10,2,40,-60,0,-10,2,-60,25,16,5,-4,-20,12,68,-7,-6,4,23,1,17,-5,-24,-4,-18,-3,24,7,17,38,-10,-25,-46,-19,-12,-28,23,-127,26,35,11,15,13,-2,-8,-39,42,-8,-3,11,23,19,7,9,34,59,19,-18,-27,-17,15,-19,23,10,47,-8,-26,30,8,8,-13,-3,41,-26,2,45,20,-20,25,10,-40,-5,-19,12,-13,10,

/\* [7][0][][] \*/ 16,8,33,-29,0,7,-14,-33,1,-10,-7,-16,-5,-90,29,10,-4,-29,-14,-5,14,-13,-38,-9,-20,-11,-3,-20,13,5,4,3,-6,6,-2,-19,5,-7,-29,29,3,-10,-27,30,-7,11,-17,67,4,22,-18,-21,-23,4,-6,-44,38,4,4,-2,-2,8,-12,-8,-15,34,11,-29,-11,-12,5,41,0,-8,-10,5,46,4,0,-17,10,-31,32,6,-33,127,-3,-4,-38,-18,-1,19,-21,4,-3,0,

/\* [8][0][][] \*/ -56,34,-43,11,48,-18,-5,38,7,61,-49,-2,8,40,73,-15,-13,26,-70,11,10,35,-34,8,-30,-27,25,10,50,-97,-6,-7,37,6,17,4,-23,31,-33,48,19,10,17,-127,-33,3,38,-22,-54,21,10,62,-106,74,19,-3,0,-19,-26,-21,3,-30,1,21,-26,34,40,-12,0,9,30,77,-31,29,27,35,-5,-84,-1,-36,-12,-17,32,7,-64,2,13,-32,6,30,-22,2,40,-18,-7,-12,

/\* [9][0][][] \*/ 22,-54,-26,-30,24,15,-9,28,32,-17,-9,4,-56,-18,10,25,-13,9,41,-27,-30,5,-23,-48,43,7,39,-20,-39,-40,-27,-7,-8,12,15,28,-1,37,-35,14,16,7,-14,-87,13,-3,30,-127,-43,-6,-1,24,-23,8,-7,1,-49,23,-17,9,6,13,21,25,12,123,43,46,16,-2,5,-30,-24,-5,-43,14,67,-9,7,17,-14,8,19,-42,-4,19,-9,-1,-59,28,-31,-8,8,3,11,-12,

/\* [10][0][][] \*/ 30,49,-28,26,62,-113,-5,-33,0,45,34,-17,-13,17,10,-80,14,42,-19,51,-88,-23,-14,-25,-76,-19,-48,51,10,3,8,51,-5,-22,39,-6,85,-26,-11,-4,-18,65,41,-23,-60,-15,-43,6,-15,-15,-39,-9,-22,-15,5,-10,14,26,-30,-11,-3,12,-6,20,3,42,61,28,19,-1,37,59,-56,-19,-59,-20,4,28,-22,15,24,-1,-75,-37,52,-47,92,-8,-23,6,18,-17,77,38,-16,-127,

/\* [11][0][][] \*/ 49,36,-22,29,-2,7,49,11,-10,-20,-26,-21,-2,8,36,2,-7,-12,-96,-8,-3,20,5,-5,-37,-22,-13,3,6,14,0,-6,1,11,8,31,6,16,-3,22,10,27,63,-100,31,-35,1,-5,-127,14,13,-50,23,19,-15,14,-22,-10,12,2,20,-12,-10,8,-46,53,-15,12,24,-5,29,13,-45,-28,11,30,0,-33,-50,5,12,-5,96,46,10,-9,-16,-5,8,20,-14,4,1,11,-17,10,

/\* [12][0][][] \*/ -37,0,12,4,-21,30,9,10,127,9,-19,14,-3,12,-32,-45,23,23,9,72,52,41,-6,-19,11,4,15,-32,-114,13,27,-49,5,1,-13,-25,12,-5,38,-16,-46,-24,2,12,7,35,12,2,13,0,-39,14,7,2,13,-11,69,-24,10,10,15,-17,6,-14,-6,37,-41,-2,4,10,-21,-42,13,7,-75,12,25,-9,11,-70,-18,4,49,42,-19,-21,-24,-30,6,25,4,49,-13,-21,18,15,

/\* [13][0][][] \*/ 29,-10,-28,23,24,2,3,-4,-7,4,12,0,14,-15,8,-11,19,-65,-34,13,-9,-21,6,1,3,11,-4,-2,-9,-26,-2,17,-4,5,9,38,19,-1,2,16,-6,11,17,35,13,12,-13,-127,23,-1,2,10,-12,-5,13,8,-5,14,23,14,7,14,-5,6,-9,-21,-20,-52,0,8,-7,-27,-9,38,-7,15,-24,2,14,-1,24,-9,-17,17,14,27,9,14,-1,0,19,-5,1,15,-13,-10,

/\* [14][0][][] \*/ -117,-64,17,45,25,-20,-32,8,-12,16,-2,11,70,1,-127,-3,81,-23,-10,43,-8,-10,42,30,-123,17,37,-5,-36,-80,-22,7,-1,6,30,-69,39,38,2,30,16,39,-6,-47,21,117,79,53,-111,52,6,43,-47,-1,38,0,13,23,72,68,42,57,13,52,5,-18,-51,42,-6,88,-47,-79,-73,7,0,53,-11,39,97,-4,66,14,34,25,52,22,51,42,42,52,74,2,-8,40,30,-36,

/\* [15][0][][] \*/ 84,-29,28,12,-14,80,24,-20,-12,62,6,11,-84,-19,-10,36,15,47,54,-23,6,25,-91,69,14,2,38,-5,22,52,-71,-90,3,-37,42,-16,34,-39,-65,-114,-4,26,10,33,-2,49,-23,-42,99,-45,-42,34,51,-114,57,-116,-1,-54,23,22,-127,-2,9,13,80,-14,-66,41,-30,71,-19,20,59,19,13,-3,16,-61,68,-10,-7,2,53,85,-22,-11,-3,-64,5,11,76,-6,-66,64,23,-14,

};

const TfArray<4, int> tensor\_dimension23 = { 4, { 1,3,3,48 } };

const TfArray<48, float> quant23\_scale = { 48, { 0.0028344567399471998, 0.0015409685438498855, 0.0086535261943936348, 0.0025573205202817917, 0.0049054129049181938, 0.0028669263701885939, 0.0074468757957220078, 0.004321360494941473, 0.0040555661544203758, 0.0016876173904165626, 0.0024430209305137396, 0.0055398130789399147, 0.0018833281937986612, 0.00185672368388623, 0.0066394438035786152, 0.0098067140206694603, 0.0083865365013480186, 0.0033648614771664143, 0.0044543137773871422, 0.0071319527924060822, 0.0046744914725422859, 0.01140966359525919, 0.004236874170601368, 0.006326745729893446, 0.0054733259603381157, 0.003714216873049736, 0.0011716821463778615, 0.0030305020045489073, 0.0042068450711667538, 0.0057273912243545055, 0.0041109221056103706, 0.0080292439088225365, 0.0023848845157772303, 0.0058926478959619999, 0.0012408687034621835, 0.0025784182362258434, 0.005969176534563303, 0.0042653433047235012, 0.0043354579247534275, 0.0057292315177619457, 0.0051813861355185509, 0.0038095219060778618, 0.0039908797480165958, 0.0034606803674250841, 0.0013780611334368587, 0.0067215678282082081, 0.0040948945097625256, 0.0014434272889047861, } };

const TfArray<4, int> tensor\_dimension43 = { 4, { 16,3,3,3 } };

const TfArray<16, float> quant43\_scale = { 16, { 3.3410194077987398e-07, 0.090170092880725861, 0.10194491595029831, 0.021601872518658638, 0.0072081638500094414, 0.10807455331087112, 0.038361717015504837, 0.15707163512706757, 0.0019356327829882503, 0.0279275793582201, 0.07762957364320755, 0.0051953857764601707, 2.9411381774480105e-07, 0.067311815917491913, 0.027543146163225174, 3.1706054670621597e-08, } };

const TfArray<16, int> quant43\_zero = { 16, { 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 } };

const TfLiteAffineQuantization quant43 = { (TfLiteFloatArray\*)&quant43\_scale, (TfLiteIntArray\*)&quant43\_zero, 0 };

const TfArray<4, int> tensor\_dimension44 = { 4, { 1,48,48,16 } };

const TfArray<1, float> quant44\_scale = { 1, { 0.023529412224888802, } };

const TfArray<1, int> quant44\_zero = { 1, { -128 } };

const TfLiteAffineQuantization quant44 = { (TfLiteFloatArray\*)&quant44\_scale, (TfLiteIntArray\*)&quant44\_zero, 0 };

const TfArray<4, int> tensor\_dimension45 = { 4, { 1,48,48,16 } };

const TfArray<1, float> quant45\_scale = { 1, { 0.023529412224888802, } };

const TfArray<1, int> quant45\_zero = { 1, { -128 } };

const TfLiteAffineQuantization quant45 = { (TfLiteFloatArray\*)&quant45\_scale, (TfLiteIntArray\*)&quant45\_zero, 0 };

const TfArray<4, int> tensor\_dimension46 = { 4, { 1,48,48,8 } };

const TfArray<1, float> quant46\_scale = { 1, { 0.45456719398498535, } };

const TfArray<1, int> quant46\_zero = { 1, { -8 } };

const TfLiteAffineQuantization quant46 = { (TfLiteFloatArray\*)&quant46\_scale,

const TfLiteDepthwiseConvParams opdata13 = { kTfLitePaddingValid, 2,2, 1, kTfLiteActRelu6, 1,1 };

const TfArray<3, int> inputs13 = { 3, { 56,23,22 } };

const TfArray<1, int> outputs13 = { 1, { 57 } };

const TfLiteConvParams opdata14 = { kTfLitePaddingSame, 1,1, kTfLiteActNone, 1,1 };

const TfArray<3, int> inputs14 = { 3, { 57,21,20 } };

const TfArray<1, int> outputs14 = { 1, { 58 } };

const TfLiteConvParams opdata15 = { kTfLitePaddingSame, 1,1, kTfLiteActRelu6, 1,1 };

const TfArray<3, int> inputs15 = { 3, { 58,19,18 } };

const TfArray<1, int> outputs15 = { 1, { 59 } };

const TfLiteDepthwiseConvParams opdata16 = { kTfLitePaddingSame, 1,1, 1, kTfLiteActRelu6, 1,1 };

const TfArray<3, int> inputs16 = { 3, { 59,17,16 } };

const TfArray<1, int> outputs16 = { 1, { 60 } };

const TfLiteConvParams opdata17 = { kTfLitePaddingSame, 1,1, kTfLiteActNone, 1,1 };

const TfArray<3, int> inputs17 = { 3, { 60,15,14 } };

const TfArray<1, int> outputs17 = { 1, { 61 } };

const TfLiteAddParams opdata18 = { kTfLiteActNone };

const TfArray<2, int> inputs18 = { 2, { 58,61 } };

const TfArray<1, int> outputs18 = { 1, { 62 } };

const TfLiteConvParams opdata19 = { kTfLitePaddingSame, 1,1, kTfLiteActRelu6, 1,1 };

const TfArray<3, int> inputs19 = { 3, { 62,13,12 } };

const TfArray<1, int> outputs19 = { 1, { 63 } };

const TfArray<3, int> inputs23 = { 3, { 66,7,6 } };

const TfArray<1, int> outputs23 = { 1, { 67 } };

const TfLiteConvParams opdata24 = { kTfLitePaddingValid, 1,1, kTfLiteActRelu, 1,1 };

const TfArray<3, int> inputs24 = { 3, { 67,5,4 } };

const TfArray<1, int> outputs24 = { 1, { 68 } };

const TfLiteSoftmaxParams opdata26 = { 1 };

const TfArray<1, int> inputs26 = { 1, { 69 } };

const TfArray<1, int> outputs26 = { 1, { 70 } };

};

TensorInfo\_t tensorData[] = {

{ kTfLiteArenaRw, kTfLiteInt8, (int32\_t\*)(tensor\_arena + 0), (TfLiteIntArray\*)&g0::tensor\_dimension0, 27648, {kTfLiteAffineQuantization, const\_cast<void\*>(static\_cast<const void\*>(&g0::quant0))}, },

{ kTfLiteMmapRo, kTfLiteInt32, (int32\_t\*)g0::tensor\_data1, (TfLiteIntArray\*)&g0::tensor\_dimension1, 32, {kTfLiteNoQuantization, nullptr}, },

{ kTfLiteMmapRo, kTfLiteInt32, (int32\_t\*)g0::tensor\_data2, (TfLiteIntArray\*)&g0::tensor\_dimension2, 8, {kTfLiteAffineQuantization, const\_cast<void\*>(static\_cast<const void\*>(&g0::quant2))}, },

{ kTfLiteMmapRo, kTfLiteInt8, (int32\_t\*)g0::tensor\_data3, (TfLiteIntArray\*)&g0::tensor\_dimension3, 64, {kTfLiteAffineQuantization, const\_cast<void\*>(static\_cast<const void\*>(&g0::quant3))}, },

{ kTfLiteMmapRo, kTfLiteInt32, (int32\_t\*)g0::tensor\_data4, (TfLiteIntArray\*)&g0::tensor\_dimension4, 128, {kTfLiteAffineQuantization, const\_cast<void\*>(static\_cast<const void\*>(&g0::quant4))}, },

{ kTfLiteMmapRo, kTfLiteInt8, (int32\_t\*)g0::tensor\_data5, (TfLiteIntArray\*)&g0::tensor\_dimension5, 3072, {kTfLiteAffineQuantization, const\_cast<void\*>(static\_cast<const void\*>(&g0::quant5))}, },

{ kTfLiteMmapRo, kTfLiteInt32, (int32\_t\*)g0::tensor\_data6, (TfLiteIntArray\*)&g0::tensor\_dimension6, 384, {kTfLiteAffineQuantization, const\_cast<void\*>(static\_cast<const void\*>(&g0::quant6))}, },

{ kTfLiteMmapRo, kTfLiteInt8, (int32\_t\*)g0::tensor\_data7, (TfLiteIntArray\*)&g0::tensor\_dimension7, 1536, {kTfLiteAffineQuantization, const\_cast<void\*>(static\_cast<const void\*>(&g0::quant7))}, },

{ kTfLiteMmapRo, kTfLiteInt32, (int32\_t\*)g0::tensor\_data8, (TfLiteIntArray\*)&g0::tensor\_dimension8, 64, {kTfLiteAffineQuantization, const\_cast<void\*>(static\_cast<const void\*>(&g0::quant8))}, },

{ kTfLiteMmapRo, kTfLiteInt8, (int32\_t\*)g0::tensor\_data9, (TfLiteIntArray\*)&g0::tensor\_dimension9, 1536, {kTfLiteAffineQuantization, const\_cast<void\*>(static\_cast<const void\*>(&g0::quant9))}, },

{ kTfLiteMmapRo, kTfLiteInt32, (int32\_t\*)g0::tensor\_data10, (TfLiteIntArray\*)&g0::tensor\_dimension10, 384, {kTfLiteAffineQuantization, const\_cast<void\*>(static\_cast<const void\*>(&g0::quant10))}, },

{ kTfLiteMmapRo, kTfLiteInt8, (int32\_t\*)g0::tensor\_data11, (TfLiteIntArray\*)&g0::tensor\_dimension11, 864, {kTfLiteAffineQuantization, const\_cast<void\*>(static\_cast<const void\*>(&g0::quant11))}, },

{ kTfLiteMmapRo, kTfLiteInt32, (int32\_t\*)g0::tensor\_data12, (TfLiteIntArray\*)&g0::tensor\_dimension12, 384, {kTfLiteAffineQuantization, const\_cast<void\*>(static\_cast<const void\*>(&g0::quant12))}, },

{ kTfLiteMmapRo, kTfLiteInt8, (int32\_t\*)g0::tensor\_data13, (TfLiteIntArray\*)&g0::tensor\_dimension13, 1536, {kTfLiteAffineQuantization, const\_cast<void\*>(static\_cast<const void\*>(&g0::quant13))}, },

{ kTfLiteMmapRo, kTfLiteInt32, (int32\_t\*)g0::tensor\_data14, (TfLiteIntArray\*)&g0::tensor\_dimension14, 64, {kTfLiteAffineQuantization, const\_cast<void\*>(static\_cast<const void\*>(&g0::quant14))}, },

{ kTfLiteMmapRo, kTfLiteInt8, (int32\_t\*)g0::tensor\_data15, (TfLiteIntArray\*)&g0::tensor\_dimension15, 1536, {kTfLiteAffineQuantization, const\_cast<void\*>(static\_cast<const void\*>(&g0::quant15))}, },

{ kTfLiteMmapRo, kTfLiteInt32, (int32\_t\*)g0::tensor\_data16, (TfLiteIntArray\*)&g0::tensor\_dimension16, 384, {kTfLiteAffineQuantization, const\_cast<void\*>(static\_cast<const void\*>(&g0::quant16))}, },

{ kTfLiteMmapRo, kTfLiteInt8, (int32\_t\*)g0::tensor\_data17, (TfLiteIntArray\*)&g0::tensor\_dimension17, 864, {kTfLiteAffineQuantization, const\_cast<void\*>(static\_cast<const void\*>(&g0::quant17))}, },

{ kTfLiteMmapRo, kTfLiteInt32, (int32\_t\*)g0::tensor\_data18, (TfLiteIntArray\*)&g0::tensor\_dimension18, 384, {kTfLiteAffineQuantization, const\_cast<void\*>(static\_cast<const void\*>(&g0::quant18))}, },

{ kTfLiteMmapRo, kTfLiteInt8, (int32\_t\*)g0::tensor\_data19, (TfLiteIntArray\*)&g0::tensor\_dimension19, 1536, {kTfLiteAffineQuantization, const\_cast<void\*>(static\_cast<const void\*>(&g0::quant19))}, },

{ kTfLiteMmapRo, kTfLiteInt32, (int32\_t\*)g0::tensor\_data20, (TfLiteIntArray\*)&g0::tensor\_dimension20, 64, {kTfLiteAffineQuantization, const\_cast<void\*>(static\_cast<const void\*>(&g0::quant20))}, },

{ kTfLiteMmapRo, kTfLiteInt8, (int32\_t\*)g0::tensor\_data21, (TfLiteIntArray\*)&g0::tensor\_dimension21, 768, {kTfLiteAffineQuantization, const\_cast<void\*>(static\_cast<const void\*>(&g0::quant21))}, },

{ kTfLiteMmapRo, kTfLiteInt32, (int32\_t\*)g0::tensor\_data22, (TfLiteIntArray\*)&g0::tensor\_dimension22, 192, {kTfLiteAffineQuantization, const\_cast<void\*>(static\_cast<const void\*>(&g0::quant22))}, },

{ kTfLiteMmapRo, kTfLiteInt8, (int32\_t\*)g0::tensor\_data23, (TfLiteIntArray\*)&g0::tensor\_dimension23, 432, {kTfLiteAffineQuantization, const\_cast<void\*>(static\_cast<const void\*>(&g0::quant23))}, },

{ kTfLiteMmapRo, kTfLiteInt32, (int32\_t\*)g0::tensor\_data24, (TfLiteIntArray\*)&g0::tensor\_dimension24, 192, {kTfLiteAffineQuantization, const\_cast<void\*>(static\_cast<const void\*>(&g0::quant24))}, },

{ kTfLiteMmapRo, kTfLiteInt8, (int32\_t\*)g0::tensor\_data25, (TfLiteIntArray\*)&g0::tensor\_dimension25, 384, {kTfLiteAffineQuantization, const\_cast<void\*>(static\_cast<const void\*>(&g0::quant25))}, },

{ kTfLiteMmapRo, kTfLiteInt32, (int32\_t\*)g0::tensor\_data26, (TfLiteIntArray\*)&g0::tensor\_dimension26, 32, {kTfLiteAffineQuantization, const\_cast<void\*>(static\_cast<const void\*>(&g0::quant26))}, },

{ kTfLiteMmapRo, kTfLiteInt8, (int32\_t\*)g0::tensor\_data27, (TfLiteIntArray\*)&g0::tensor\_dimension27, 384, {kTfLiteAffineQuantization, const\_cast<void\*>(static\_cast<const void\*>(&g0::quant27))}, },

{ kTfLiteMmapRo, kTfLiteInt32, (int32\_t\*)g0::tensor\_data28, (TfLiteIntArray\*)&g0::tensor\_dimension28, 192, {kTfLiteAffineQuantization, const\_cast<void\*>(static\_cast<const void\*>(&g0::quant28))}, },

{ kTfLiteMmapRo, kTfLiteInt8, (int32\_t\*)g0::tensor\_data29, (TfLiteIntArray\*)&g0::tensor\_dimension29, 432, {kTfLiteAffineQuantization, const\_cast<void\*>(static\_cast<const void\*>(&g0::quant29))}, },

{ kTfLiteMmapRo, kTfLiteInt32, (int32\_t\*)g0::tensor\_data30, (TfLiteIntArray\*)&g0::tensor\_dimension30, 192, {kTfLiteAffineQuantization, const\_cast<void\*>(static\_cast<const void\*>(&g0::quant30))}, },

{ kTfLiteMmapRo, kTfLiteInt8, (int32\_t\*)g0::tensor\_data31, (TfLiteIntArray\*)&g0::tensor\_dimension31, 384, {kTfLiteAffineQuantization, const\_cast<void\*>(static\_cast<const void\*>(&g0::quant31))}, },

{ kTfLiteMmapRo, kTfLiteInt32, (int32\_t\*)g0::tensor\_data32, (TfLiteIntArray\*)&g0::tensor\_dimension32, 32, {kTfLiteAffineQuantization, const\_cast<void\*>(static\_cast<const void\*>(&g0::quant32))}, },

{ kTfLiteMmapRo, kTfLiteInt8, (int32\_t\*)g0::tensor\_data33, (TfLiteIntArray\*)&g0::tensor\_dimension33, 384, {kTfLiteAffineQuantization, const\_cast<void\*>(static\_cast<const void\*>(&g0::quant33))}, },

{ kTfLiteMmapRo, kTfLiteInt32, (int32\_t\*)g0::tensor\_data34, (TfLiteIntArray\*)&g0::tensor\_dimension34, 192, {kTfLiteAffineQuantization, const\_cast<void\*>(static\_cast<const void\*>(&g0::quant34))}, },

{ kTfLiteMmapRo, kTfLiteInt8, (int32\_t\*)g0::tensor\_data35, (TfLiteIntArray\*)&g0::tensor\_dimension35, 432, {kTfLiteAffineQuantization, const\_cast<void\*>(static\_cast<const void\*>(&g0::quant35))}, },

{ kTfLiteMmapRo, kTfLiteInt32, (int32\_t\*)g0::tensor\_data36, (TfLiteIntArray\*)&g0::tensor\_dimension36, 192, {kTfLiteAffineQuantization, const\_cast<void\*>(static\_cast<const void\*>(&g0::quant36))}, },

{ kTfLiteMmapRo, kTfLiteInt8, (int32\_t\*)g0::tensor\_data37, (TfLiteIntArray\*)&g0::tensor\_dimension37, 384, {kTfLiteAffineQuantization, const\_cast<void\*>(static\_cast<const void\*>(&g0::quant37))}, },

{ kTfLiteMmapRo, kTfLiteInt32, (int32\_t\*)g0::tensor\_data38, (TfLiteIntArray\*)&g0::tensor\_dimension38, 32, {kTfLiteAffineQuantization, const\_cast<void\*>(static\_cast<const void\*>(&g0::quant38))}, },

{ kTfLiteMmapRo, kTfLiteInt8, (int32\_t\*)g0::tensor\_data39, (TfLiteIntArray\*)&g0::tensor\_dimension39, 128, {kTfLiteAffineQuantization, const\_cast<void\*>(static\_cast<const void\*>(&g0::quant39))}, },

{ kTfLiteMmapRo, kTfLiteInt32, (int32\_t\*)g0::tensor\_data40, (TfLiteIntArray\*)&g0::tensor\_dimension40, 64, {kTfLiteAffineQuantization, const\_cast<void\*>(static\_cast<const void\*>(&g0::quant40))}, },

{ kTfLiteMmapRo, kTfLiteInt8, (int32\_t\*)g0::tensor\_data41, (TfLiteIntArray\*)&g0::tensor\_dimension41, 144, {kTfLiteAffineQuantization, const\_cast<void\*>(static\_cast<const void\*>(&g0::quant41))}, },

{ kTfLiteMmapRo, kTfLiteInt32, (int32\_t\*)g0::tensor\_data42, (TfLiteIntArray\*)&g0::tensor\_dimension42, 64, {kTfLiteAffineQuantization, const\_cast<void\*>(static\_cast<const void\*>(&g0::quant42))}, },

{ kTfLiteMmapRo, kTfLiteInt8, (int32\_t\*)g0::tensor\_data43, (TfLiteIntArray\*)&g0::tensor\_dimension43, 432, {kTfLiteAffineQuantization, const\_cast<void\*>(static\_cast<const void\*>(&g0::quant43))}, },

{ kTfLiteArenaRw, kTfLiteInt8, (int32\_t\*)(tensor\_arena + 36864), (TfLiteIntArray\*)&g0::tensor\_dimension44, 36864, {kTfLiteAffineQuantization, const\_cast<void\*>(static\_cast<const void\*>(&g0::quant44))}, },

{ kTfLiteArenaRw, kTfLiteInt8, (int32\_t\*)(tensor\_arena + 0), (TfLiteIntArray\*)&g0::tensor\_dimension45, 36864, {kTfLiteAffineQuantization, const\_cast<void\*>(static\_cast<const void\*>(&g0::quant45))}, },

{ kTfLiteArenaRw, kTfLiteInt8, (int32\_t\*)(tensor\_arena + 36864), (TfLiteIntArray\*)&g0::tensor\_dimension46, 18432, {kTfLiteAffineQuantization, const\_cast<void\*>(static\_cast<const void\*>(&g0::quant46))}, },

{ kTfLiteArenaRw, kTfLiteInt8, (int32\_t\*)(tensor\_arena + 115248), (TfLiteIntArray\*)&g0::tensor\_dimension47, 110592, {kTfLiteAffineQuantization, const\_cast<void\*>(static\_cast<const void\*>(&g0::quant47))}, },

{ kTfLiteArenaRw, kTfLiteInt8, (int32\_t\*)(tensor\_arena + 0), (TfLiteIntArray\*)&g0::tensor\_dimension48, 115248, {kTfLiteAffineQuantization, const\_cast<void\*>(static\_cast<const void\*>(&g0::quant48))}, },

{ kTfLiteArenaRw, kTfLiteInt8, (int32\_t\*)(tensor\_arena + 115248), (TfLiteIntArray\*)&g0::tensor\_dimension49, 27648, {kTfLiteAffineQuantization, const\_cast<void\*>(static\_cast<const void\*>(&g0::quant49))}, },

{ kTfLiteArenaRw, kTfLiteInt8, (int32\_t\*)(tensor\_arena + 55296), (TfLiteIntArray\*)&g0::tensor\_dimension50, 4608, {kTfLiteAffineQuantization, const\_cast<void\*>(static\_cast<const void\*>(&g0::quant50))}, },

{ kTfLiteArenaRw, kTfLiteInt8, (int32\_t\*)(tensor\_arena + 27648), (TfLiteIntArray\*)&g0::tensor\_dimension51, 27648, {kTfLiteAffineQuantization, const\_cast<void\*>(static\_cast<const void\*>(&g0::quant51))}, },

{ kTfLiteArenaRw, kTfLiteInt8, (int32\_t\*)(tensor\_arena + 0), (TfLiteIntArray\*)&g0::tensor\_dimension52, 27648, {kTfLiteAffineQuantization, const\_cast<void\*>(static\_cast<const void\*>(&g0::quant52))}, },

{ kTfLiteArenaRw, kTfLiteInt8, (int32\_t\*)(tensor\_arena + 27648), (TfLiteIntArray\*)&g0::tensor\_dimension53, 4608, {kTfLiteAffineQuantization, const\_cast<void\*>(static\_cast<const void\*>(&g0::quant53))}, },

{ kTfLiteArenaRw, kTfLiteInt8, (int32\_t\*)(tensor\_arena + 0), (TfLiteIntArray\*)&g0::tensor\_dimension54, 4608, {kTfLiteAffineQuantization, const\_cast<void\*>(static\_cast<const void\*>(&g0::quant54))}, },

{ kTfLiteArenaRw, kTfLiteInt8, (int32\_t\*)(tensor\_arena + 30000), (TfLiteIntArray\*)&g0::tensor\_dimension55, 27648, {kTfLiteAffineQuantization, const\_cast<void\*>(static\_cast<const void\*>(&g0::quant55))}, },

{ kTfLiteArenaRw, kTfLiteInt8, (int32\_t\*)(tensor\_arena + 0), (TfLiteIntArray\*)&g0::tensor\_dimension56, 30000, {kTfLiteAffineQuantization, const\_cast<void\*>(static\_cast<const void\*>(&g0::quant56))}, },

{ kTfLiteArenaRw, kTfLiteInt8, (int32\_t\*)(tensor\_arena + 30000), (TfLiteIntArray\*)&g0::tensor\_dimension57, 6912, {kTfLiteAffineQuantization, const\_cast<void\*>(static\_cast<const void\*>(&g0::quant57))}, },

{ kTfLiteArenaRw, kTfLiteInt8, (int32\_t\*)(tensor\_arena + 36912), (TfLiteIntArray\*)&g0::tensor\_dimension58, 2304, {kTfLiteAffineQuantization, const\_cast<void\*>(static\_cast<const void\*>(&g0::quant58))}, },

{ kTfLiteArenaRw, kTfLiteInt8, (int32\_t\*)(tensor\_arena + 13824), (TfLiteIntArray\*)&g0::tensor\_dimension59, 13824, {kTfLiteAffineQuantization, const\_cast<void\*>(static\_cast<const void\*>(&g0::quant59))}, },

{ kTfLiteArenaRw, kTfLiteInt8, (int32\_t\*)(tensor\_arena + 0), (TfLiteIntArray\*)&g0::tensor\_dimension60, 13824, {kTfLiteAffineQuantization, const\_cast<void\*>(static\_cast<const void\*>(&g0::quant60))}, },

{ kTfLiteArenaRw, kTfLiteInt8, (int32\_t\*)(tensor\_arena + 13824), (TfLiteIntArray\*)&g0::tensor\_dimension61, 2304, {kTfLiteAffineQuantization, const\_cast<void\*>(static\_cast<const void\*>(&g0::quant61))}, },

{ kTfLiteArenaRw, kTfLiteInt8, (int32\_t\*)(tensor\_arena + 27648), (TfLiteIntArray\*)&g0::tensor\_dimension62, 2304, {kTfLiteAffineQuantization, const\_cast<void\*>(static\_cast<const void\*>(&g0::quant62))}, },

{ kTfLiteArenaRw, kTfLiteInt8, (int32\_t\*)(tensor\_arena + 13824), (TfLiteIntArray\*)&g0::tensor\_dimension63, 13824, {kTfLiteAffineQuantization, const\_cast<void\*>(static\_cast<const void\*>(&g0::quant63))}, },

{ kTfLiteArenaRw, kTfLiteInt8, (int32\_t\*)(tensor\_arena + 0), (TfLiteIntArray\*)&g0::tensor\_dimension64, 13824, {kTfLiteAffineQuantization, const\_cast<void\*>(static\_cast<const void\*>(&g0::quant64))}, },

{ kTfLiteArenaRw, kTfLiteInt8, (int32\_t\*)(tensor\_arena + 16128), (TfLiteIntArray\*)&g0::tensor\_dimension65, 2304, {kTfLiteAffineQuantization, const\_cast<void\*>(static\_cast<const void\*>(&g0::quant65))}, },

{ kTfLiteArenaRw, kTfLiteInt8, (int32\_t\*)(tensor\_arena + 13824), (TfLiteIntArray\*)&g0::tensor\_dimension66, 2304, {kTfLiteAffineQuantization, const\_cast<void\*>(static\_cast<const void\*>(&g0::quant66))}, },

{ kTfLiteArenaRw, kTfLiteInt8, (int32\_t\*)(tensor\_arena + 0), (TfLiteIntArray\*)&g0::tensor\_dimension67, 13824, {kTfLiteAffineQuantization, const\_cast<void\*>(static\_cast<const void\*>(&g0::quant67))}, },

{ kTfLiteArenaRw, kTfLiteInt8, (int32\_t\*)(tensor\_arena + 13824), (TfLiteIntArray\*)&g0::tensor\_dimension68, 4608, {kTfLiteAffineQuantization, const\_cast<void\*>(static\_cast<const void\*>(&g0::quant68))}, },

{ kTfLiteArenaRw, kTfLiteInt8, (int32\_t\*)(tensor\_arena + 288), (TfLiteIntArray\*)&g0::tensor\_dimension69, 288, {kTfLiteAffineQuantization, const\_cast<void\*>(static\_cast<const void\*>(&g0::quant69))}, },

{ kTfLiteArenaRw, kTfLiteInt8, (int32\_t\*)(tensor\_arena + 0), (TfLiteIntArray\*)&g0::tensor\_dimension70, 288, {kTfLiteAffineQuantization, const\_cast<void\*>(static\_cast<const void\*>(&g0::quant70))}, },

};

TfLiteNode tflNodes[27] = {

{ (TfLiteIntArray\*)&g0::inputs0, (TfLiteIntArray\*)&g0::outputs0, (TfLiteIntArray\*)&g0::inputs0, nullptr, const\_cast<void\*>(static\_cast<const void\*>(&g0::opdata0)), nullptr, 0, },

{ (TfLiteIntArray\*)&g0::inputs1, (TfLiteIntArray\*)&g0::outputs1, (TfLiteIntArray\*)&g0::inputs1, nullptr, const\_cast<void\*>(static\_cast<const void\*>(&g0::opdata1)), nullptr, 0, },

{ (TfLiteIntArray\*)&g0::inputs2, (TfLiteIntArray\*)&g0::outputs2, (TfLiteIntArray\*)&g0::inputs2, nullptr, const\_cast<void\*>(static\_cast<const void\*>(&g0::opdata2)), nullptr, 0, },

{ (TfLiteIntArray\*)&g0::inputs3, (TfLiteIntArray\*)&g0::outputs3, (TfLiteIntArray\*)&g0::inputs3, nullptr, const\_cast<void\*>(static\_cast<const void\*>(&g0::opdata3)), nullptr, 0, },

{ (TfLiteIntArray\*)&g0::inputs4, (TfLiteIntArray\*)&g0::outputs4, (TfLiteIntArray\*)&g0::inputs4, nullptr, nullptr, nullptr, 0, },

{ (TfLiteIntArray\*)&g0::inputs5, (TfLiteIntArray\*)&g0::outputs5, (TfLiteIntArray\*)&g0::inputs5, nullptr, const\_cast<void\*>(static\_cast<const void\*>(&g0::opdata5)), nullptr, 0, },

{ (TfLiteIntArray\*)&g0::inputs6, (TfLiteIntArray\*)&g0::outputs6, (TfLiteIntArray\*)&g0::inputs6, nullptr, const\_cast<void\*>(static\_cast<const void\*>(&g0::opdata6)), nullptr, 0, },

{ (TfLiteIntArray\*)&g0::inputs7, (TfLiteIntArray\*)&g0::outputs7, (TfLiteIntArray\*)&g0::inputs7, nullptr, const\_cast<void\*>(static\_cast<const void\*>(&g0::opdata7)), nullptr, 0, },

{ (TfLiteIntArray\*)&g0::inputs8, (TfLiteIntArray\*)&g0::outputs8, (TfLiteIntArray\*)&g0::inputs8, nullptr, const\_cast<void\*>(static\_cast<const void\*>(&g0::opdata8)), nullptr, 0, },

{ (TfLiteIntArray\*)&g0::inputs9, (TfLiteIntArray\*)&g0::outputs9, (TfLiteIntArray\*)&g0::inputs9, nullptr, const\_cast<void\*>(static\_cast<const void\*>(&g0::opdata9)), nullptr, 0, },

{ (TfLiteIntArray\*)&g0::inputs10, (TfLiteIntArray\*)&g0::outputs10, (TfLiteIntArray\*)&g0::inputs10, nullptr, const\_cast<void\*>(static\_cast<const void\*>(&g0::opdata10)), nullptr, 0, },

{ (TfLiteIntArray\*)&g0::inputs11, (TfLiteIntArray\*)&g0::outputs11, (TfLiteIntArray\*)&g0::inputs11, nullptr, const\_cast<void\*>(static\_cast<const void\*>(&g0::opdata11)), nullptr, 0, },

{ (TfLiteIntArray\*)&g0::inputs12, (TfLiteIntArray\*)&g0::outputs12, (TfLiteIntArray\*)&g0::inputs12, nullptr, nullptr, nullptr, 0, },

{ (TfLiteIntArray\*)&g0::inputs13, (TfLiteIntArray\*)&g0::outputs13, (TfLiteIntArray\*)&g0::inputs13, nullptr, const\_cast<void\*>(static\_cast<const void\*>(&g0::opdata13)), nullptr, 0, },

{ (TfLiteIntArray\*)&g0::inputs14, (TfLiteIntArray\*)&g0::outputs14, (TfLiteIntArray\*)&g0::inputs14, nullptr, const\_cast<void\*>(static\_cast<const void\*>(&g0::opdata14)), nullptr, 0, },

{ (TfLiteIntArray\*)&g0::inputs15, (TfLiteIntArray\*)&g0::outputs15, (TfLiteIntArray\*)&g0::inputs15, nullptr, const\_cast<void\*>(static\_cast<const void\*>(&g0::opdata15)), nullptr, 0, },

{ (TfLiteIntArray\*)&g0::inputs16, (TfLiteIntArray\*)&g0::outputs16, (TfLiteIntArray\*)&g0::inputs16, nullptr, const\_cast<void\*>(static\_cast<const void\*>(&g0::opdata16)), nullptr, 0, },

{ (TfLiteIntArray\*)&g0::inputs17, (TfLiteIntArray\*)&g0::outputs17, (TfLiteIntArray\*)&g0::inputs17, nullptr, const\_cast<void\*>(static\_cast<const void\*>(&g0::opdata17)), nullptr, 0, },

{ (TfLiteIntArray\*)&g0::inputs18, (TfLiteIntArray\*)&g0::outputs18, (TfLiteIntArray\*)&g0::inputs18, nullptr, const\_cast<void\*>(static\_cast<const void\*>(&g0::opdata18)), nullptr, 0, },

{ (TfLiteIntArray\*)&g0::inputs19, (TfLiteIntArray\*)&g0::outputs19, (TfLiteIntArray\*)&g0::inputs19, nullptr, const\_cast<void\*>(static\_cast<const void\*>(&g0::opdata19)), nullptr, 0, },

{ (TfLiteIntArray\*)&g0::inputs20, (TfLiteIntArray\*)&g0::outputs20, (TfLiteIntArray\*)&g0::inputs20, nullptr, const\_cast<void\*>(static\_cast<const void\*>(&g0::opdata20)), nullptr, 0, },

{ (TfLiteIntArray\*)&g0::inputs21, (TfLiteIntArray\*)&g0::outputs21, (TfLiteIntArray\*)&g0::inputs21, nullptr, const\_cast<void\*>(static\_cast<const void\*>(&g0::opdata21)), nullptr, 0, },

{ (TfLiteIntArray\*)&g0::inputs22, (TfLiteIntArray\*)&g0::outputs22, (TfLiteIntArray\*)&g0::inputs22, nullptr, const\_cast<void\*>(static\_cast<const void\*>(&g0::opdata22)), nullptr, 0, },

{ (TfLiteIntArray\*)&g0::inputs23, (TfLiteIntArray\*)&g0::outputs23, (TfLiteIntArray\*)&g0::inputs23, nullptr, const\_cast<void\*>(static\_cast<const void\*>(&g0::opdata23)), nullptr, 0, },

{ (TfLiteIntArray\*)&g0::inputs24, (TfLiteIntArray\*)&g0::outputs24, (TfLiteIntArray\*)&g0::inputs24, nullptr, const\_cast<void\*>(static\_cast<const void\*>(&g0::opdata24)), nullptr, 0, },

{ (TfLiteIntArray\*)&g0::inputs25, (TfLiteIntArray\*)&g0::outputs25, (TfLiteIntArray\*)&g0::inputs25, nullptr, const\_cast<void\*>(static\_cast<const void\*>(&g0::opdata25)), nullptr, 0, },

{ (TfLiteIntArray\*)&g0::inputs26, (TfLiteIntArray\*)&g0::outputs26, (TfLiteIntArray\*)&g0::inputs26, nullptr, const\_cast<void\*>(static\_cast<const void\*>(&g0::opdata26)), nullptr, 0, },

};

#endif

used\_operators\_e used\_ops[] =

{OP\_CONV\_2D, OP\_DEPTHWISE\_CONV\_2D, OP\_CONV\_2D, OP\_CONV\_2D, OP\_PAD, OP\_DEPTHWISE\_CONV\_2D, OP\_CONV\_2D, OP\_CONV\_2D, OP\_DEPTHWISE\_CONV\_2D, OP\_CONV\_2D, OP\_ADD, OP\_CONV\_2D, OP\_PAD, OP\_DEPTHWISE\_CONV\_2D, OP\_CONV\_2D, OP\_CONV\_2D, OP\_DEPTHWISE\_CONV\_2D, OP\_CONV\_2D, OP\_ADD, OP\_CONV\_2D, OP\_DEPTHWISE\_CONV\_2D, OP\_CONV\_2D, OP\_ADD, OP\_CONV\_2D, OP\_CONV\_2D, OP\_CONV\_2D, OP\_SOFTMAX, };

// Indices into tflTensors and tflNodes for subgraphs

const size\_t tflTensors\_subgraph\_index[] = {0, 71, };

const size\_t tflNodes\_subgraph\_index[] = {0, 27, };

// Input/output tensors

static const int in\_tensor\_indices[] = {

0,

};

static const int out\_tensor\_indices[] = {

70,

};

size\_t current\_subgraph\_index = 0;

static void init\_tflite\_tensor(size\_t i, TfLiteTensor \*tensor) {

tensor->type = tensorData[i].type;

tensor->is\_variable = false;

#if defined(EI\_CLASSIFIER\_ALLOCATION\_HEAP)

tensor->allocation\_type = tensorData[i].allocation\_type;

#else

tensor->allocation\_type = (tensor\_arena <= tensorData[i].data && tensorData[i].data < tensor\_arena + kTensorArenaSize) ? kTfLiteArenaRw : kTfLiteMmapRo;

#endif

tensor->bytes = tensorData[i].bytes;

tensor->dims = tensorData[i].dims;

#if defined(EI\_CLASSIFIER\_ALLOCATION\_HEAP)

if(tensor->allocation\_type == kTfLiteArenaRw){

uint8\_t\* start = (uint8\_t\*) ((uintptr\_t)tensorData[i].data + (uintptr\_t) tensor\_arena);

tensor->data.data = start;

}

else {

tensor->data.data = tensorData[i].data;

}

#else

tensor->data.data = tensorData[i].data;

#endif // EI\_CLASSIFIER\_ALLOCATION\_HEAP

tensor->quantization = tensorData[i].quantization;

if (tensor->quantization.type == kTfLiteAffineQuantization) {

TfLiteAffineQuantization const\* quant = ((TfLiteAffineQuantization const\*)(tensorData[i].quantization.params));

tensor->params.scale = quant->scale->data[0];

tensor->params.zero\_point = quant->zero\_point->data[0];

}

}

static void init\_tflite\_eval\_tensor(int i, TfLiteEvalTensor \*tensor) {

tensor->type = tensorData[i].type;

tensor->dims = tensorData[i].dims;

#if defined(EI\_CLASSIFIER\_ALLOCATION\_HEAP)

auto allocation\_type = tensorData[i].allocation\_type;

if(allocation\_type == kTfLiteArenaRw) {

uint8\_t\* start = (uint8\_t\*) ((uintptr\_t)tensorData[i].data + (uintptr\_t) tensor\_arena);

tensor->data.data = start;

}

else {

tensor->data.data = tensorData[i].data;

}

#else

tensor->data.data = tensorData[i].data;

#endif // EI\_CLASSIFIER\_ALLOCATION\_HEAP

}

static void\* overflow\_buffers[EI\_MAX\_OVERFLOW\_BUFFER\_COUNT];

static size\_t overflow\_buffers\_ix = 0;

static void \* AllocatePersistentBufferImpl(struct TfLiteContext\* ctx,

size\_t bytes) {

void \*ptr;

uint32\_t align\_bytes = (bytes % 16) ? 16 - (bytes % 16) : 0;

if (current\_location - (bytes + align\_bytes) < tensor\_boundary) {

if (overflow\_buffers\_ix > EI\_MAX\_OVERFLOW\_BUFFER\_COUNT - 1) {

ei\_printf("ERR: Failed to allocate persistent buffer of size %d, does not fit in tensor arena and reached EI\_MAX\_OVERFLOW\_BUFFER\_COUNT\n",

(int)bytes);

return NULL;

}

// OK, this will look super weird, but.... we have CMSIS-NN buffers which

// we cannot calculate beforehand easily.

ptr = ei\_calloc(bytes, 1);

if (ptr == NULL) {

ei\_printf("ERR: Failed to allocate persistent buffer of size %d\n", (int)bytes);

return NULL;

}

overflow\_buffers[overflow\_buffers\_ix++] = ptr;

return ptr;

}

current\_location -= bytes;

// align to the left aligned boundary of 16 bytes

current\_location -= 15; // for alignment

current\_location += 16 - ((uintptr\_t)(current\_location) & 15);

ptr = current\_location;

memset(ptr, 0, bytes);

return ptr;

}

typedef struct {

size\_t bytes;

void \*ptr;

} scratch\_buffer\_t;

static scratch\_buffer\_t scratch\_buffers[EI\_MAX\_SCRATCH\_BUFFER\_COUNT];

static size\_t scratch\_buffers\_ix = 0;

static TfLiteStatus RequestScratchBufferInArenaImpl(struct TfLiteContext\* ctx, size\_t bytes,

int\* buffer\_idx) {

if (scratch\_buffers\_ix > EI\_MAX\_SCRATCH\_BUFFER\_COUNT - 1) {

ei\_printf("ERR: Failed to allocate scratch buffer of size %d, reached EI\_MAX\_SCRATCH\_BUFFER\_COUNT\n",

(int)bytes);

return kTfLiteError;

}

scratch\_buffer\_t b;

b.bytes = bytes;

b.ptr = AllocatePersistentBufferImpl(ctx, b.bytes);

if (!b.ptr) {

ei\_printf("ERR: Failed to allocate scratch buffer of size %d\n",

(int)bytes);

return kTfLiteError;

}

scratch\_buffers[scratch\_buffers\_ix] = b;

\*buffer\_idx = scratch\_buffers\_ix;

scratch\_buffers\_ix++;

return kTfLiteOk;

}

static void\* GetScratchBufferImpl(struct TfLiteContext\* ctx, int buffer\_idx) {

if (buffer\_idx > (int)scratch\_buffers\_ix) {

return NULL;

}

return scratch\_buffers[buffer\_idx].ptr;

}

static const uint16\_t TENSOR\_IX\_UNUSED = 0x7FFF;

static void ResetTensors() {

for (size\_t ix = 0; ix < MAX\_TFL\_TENSOR\_COUNT; ix++) {

tflTensors[ix].index = TENSOR\_IX\_UNUSED;

}

for (size\_t ix = 0; ix < MAX\_TFL\_EVAL\_COUNT; ix++) {

tflEvalTensors[ix].index = TENSOR\_IX\_UNUSED;

}

}

static TfLiteTensor\* GetTensorImpl(const struct TfLiteContext\* context,

int tensor\_idx) {

tensor\_idx = tflTensors\_subgraph\_index[current\_subgraph\_index] + tensor\_idx;

for (size\_t ix = 0; ix < MAX\_TFL\_TENSOR\_COUNT; ix++) {

// already used? OK!

if (tflTensors[ix].index == tensor\_idx) {

return &tflTensors[ix].tensor;

}

// passed all the ones we've used, so end of the list?

if (tflTensors[ix].index == TENSOR\_IX\_UNUSED) {

// init the tensor

init\_tflite\_tensor(tensor\_idx, &tflTensors[ix].tensor);

tflTensors[ix].index = tensor\_idx;

return &tflTensors[ix].tensor;

}

}

ei\_printf("ERR: GetTensor called beyond MAX\_TFL\_TENSOR\_COUNT (%d)\n", MAX\_TFL\_TENSOR\_COUNT);

return nullptr;

}

static TfLiteEvalTensor\* GetEvalTensorImpl(const struct TfLiteContext\* context,

int tensor\_idx) {

tensor\_idx = tflTensors\_subgraph\_index[current\_subgraph\_index] + tensor\_idx;

for (size\_t ix = 0; ix < MAX\_TFL\_EVAL\_COUNT; ix++) {

// already used? OK!

if (tflEvalTensors[ix].index == tensor\_idx) {

return &tflEvalTensors[ix].tensor;

}

// passed all the ones we've used, so end of the list?

if (tflEvalTensors[ix].index == TENSOR\_IX\_UNUSED) {

// init the tensor

init\_tflite\_eval\_tensor(tensor\_idx, &tflEvalTensors[ix].tensor);

tflEvalTensors[ix].index = tensor\_idx;

return &tflEvalTensors[ix].tensor;

}

}

ei\_printf("ERR: GetTensor called beyond MAX\_TFL\_EVAL\_COUNT (%d)\n", (int)MAX\_TFL\_EVAL\_COUNT);

return nullptr;

}

class EonMicroContext : public MicroContext {

public:

EonMicroContext(): MicroContext(nullptr, nullptr, nullptr) { }

void\* AllocatePersistentBuffer(size\_t bytes) {

return AllocatePersistentBufferImpl(nullptr, bytes);

}

TfLiteStatus RequestScratchBufferInArena(size\_t bytes,

int\* buffer\_index) {

return RequestScratchBufferInArenaImpl(nullptr, bytes, buffer\_index);

}

void\* GetScratchBuffer(int buffer\_index) {

return GetScratchBufferImpl(nullptr, buffer\_index);

}

TfLiteTensor\* AllocateTempTfLiteTensor(int tensor\_index) {

return GetTensorImpl(nullptr, tensor\_index);

}

void DeallocateTempTfLiteTensor(TfLiteTensor\* tensor) {

return;

}

bool IsAllTempTfLiteTensorDeallocated() {

return true;

}

TfLiteEvalTensor\* GetEvalTensor(int tensor\_index) {

return GetEvalTensorImpl(nullptr, tensor\_index);

}

};

} // namespace

memset(tensor\_arena, 0, kTensorArenaSize);

#endif

tensor\_boundary = tensor\_arena;

current\_location = tensor\_arena + kTensorArenaSize;

EonMicroContext micro\_context\_;

// Set microcontext as the context ptr

ctx.impl\_ = static\_cast<void\*>(&micro\_context\_);

// Setup tflitecontext functions

ctx.AllocatePersistentBuffer = &AllocatePersistentBufferImpl;

ctx.RequestScratchBufferInArena = &RequestScratchBufferInArenaImpl;

ctx.GetScratchBuffer = &GetScratchBufferImpl;

ctx.GetTensor = &GetTensorImpl;

ctx.GetEvalTensor = &GetEvalTensorImpl;

ctx.ReportError = &MicroContextReportOpError;

ctx.tensors\_size = 71;

for (size\_t i = 0; i < 71; ++i) {

TfLiteTensor tensor;

init\_tflite\_tensor(i, &tensor);

if (tensor.allocation\_type == kTfLiteArenaRw) {

auto data\_end\_ptr = (uint8\_t\*)tensor.data.data + tensorData[i].bytes;

if (data\_end\_ptr > tensor\_boundary) {

tensor\_boundary = data\_end\_ptr;

}

}

}

if (tensor\_boundary > current\_location /\* end of arena size \*/) {

ei\_printf("ERR: tensor arena is too small, does not fit model - even without scratch buffers\n");

return kTfLiteError;

}

registrations[OP\_CONV\_2D] = Register\_CONV\_2D();

registrations[OP\_DEPTHWISE\_CONV\_2D] = Register\_DEPTHWISE\_CONV\_2D();

registrations[OP\_PAD] = Register\_PAD();

registrations[OP\_ADD] = Register\_ADD();

registrations[OP\_SOFTMAX] = Register\_SOFTMAX();

for (size\_t g = 0; g < 1; ++g) {

current\_subgraph\_index = g;

for(size\_t i = tflNodes\_subgraph\_index[g]; i < tflNodes\_subgraph\_index[g+1]; ++i) {

if (registrations[used\_ops[i]].init) {

tflNodes[i].user\_data = registrations[used\_ops[i]].init(&ctx, (const char\*)tflNodes[i].builtin\_data, 0);

}

}

}

current\_subgraph\_index = 0;

for(size\_t g = 0; g < 1; ++g) {

current\_subgraph\_index = g;

for(size\_t i = tflNodes\_subgraph\_index[g]; i < tflNodes\_subgraph\_index[g+1]; ++i) {

if (registrations[used\_ops[i]].prepare) {

ResetTensors();

TfLiteStatus status = registrations[used\_ops[i]].prepare(&ctx, &tflNodes[i]);

if (status != kTfLiteOk) {

return status;

}

}

}

}

current\_subgraph\_index = 0;

return kTfLiteOk;

}

TfLiteStatus tflite\_learn\_5\_input(int index, TfLiteTensor \*tensor) {

init\_tflite\_tensor(in\_tensor\_indices[index], tensor);

return kTfLiteOk;

}

TfLiteStatus tflite\_learn\_5\_output(int index, TfLiteTensor \*tensor) {

init\_tflite\_tensor(out\_tensor\_indices[index], tensor);

return kTfLiteOk;

}

TfLiteStatus tflite\_learn\_5\_invoke() {

for (size\_t i = 0; i < 27; ++i) {

ResetTensors();

TfLiteStatus status = registrations[used\_ops[i]].invoke(&ctx, &tflNodes[i]);

#if EI\_CLASSIFIER\_PRINT\_STATE

ei\_printf("layer %lu\n", i);

ei\_printf(" inputs:\n");

for (size\_t ix = 0; ix < tflNodes[i].inputs->size; ix++) {

auto d = tensorData[tflNodes[i].inputs->data[ix]];

size\_t data\_ptr = (size\_t)d.data;

if (d.allocation\_type == kTfLiteArenaRw) {

data\_ptr = (size\_t)tensor\_arena + data\_ptr;

}

if (d.type == TfLiteType::kTfLiteInt8) {

int8\_t\* data = (int8\_t\*)data\_ptr;

ei\_printf(" %lu (%zu bytes, ptr=%p, alloc\_type=%d, type=%d): ", ix, d.bytes, data, (int)d.allocation\_type, (int)d.type);

for (size\_t jx = 0; jx < d.bytes; jx++) {

ei\_printf("%d ", data[jx]);

}

}

else {

float\* data = (float\*)data\_ptr;

ei\_printf(" %lu (%zu bytes, ptr=%p, alloc\_type=%d, type=%d): ", ix, d.bytes, data, (int)d.allocation\_type, (int)d.type);

for (size\_t jx = 0; jx < d.bytes / 4; jx++) {

ei\_printf("%f ", data[jx]);

}

}

ei\_printf("\n");

}

ei\_printf("\n");

ei\_printf(" outputs:\n");

for (size\_t ix = 0; ix < tflNodes[i].outputs->size; ix++) {

auto d = tensorData[tflNodes[i].outputs->data[ix]];

size\_t data\_ptr = (size\_t)d.data;

if (d.allocation\_type == kTfLiteArenaRw) {

data\_ptr = (size\_t)tensor\_arena + data\_ptr;

}

if (d.type == TfLiteType::kTfLiteInt8) {

int8\_t\* data = (int8\_t\*)data\_ptr;

ei\_printf(" %lu (%zu bytes, ptr=%p, alloc\_type=%d, type=%d): ", ix, d.bytes, data, (int)d.allocation\_type, (int)d.type);

for (size\_t jx = 0; jx < d.bytes; jx++) {

ei\_printf("%d ", data[jx]);

}

}

else {

float\* data = (float\*)data\_ptr;

ei\_printf(" %lu (%zu bytes, ptr=%p, alloc\_type=%d, type=%d): ", ix, d.bytes, data, (int)d.allocation\_type, (int)d.type);

for (size\_t jx = 0; jx < d.bytes / 4; jx++) {

ei\_printf("%f ", data[jx]);

}

}

ei\_printf("\n");

}

ei\_printf("\n");

#endif // EI\_CLASSIFIER\_PRINT\_STATE

if (status != kTfLiteOk) {

return status;

}

}

return kTfLiteOk;

}

TfLiteStatus tflite\_learn\_5\_reset( void (\*free\_fnc)(void\* ptr) ) {

#ifdef EI\_CLASSIFIER\_ALLOCATION\_HEAP

free\_fnc(tensor\_arena);

#endif

// scratch buffers are allocated within the arena, so just reset the counter so memory can be reused

scratch\_buffers\_ix = 0;

// overflow buffers are on the heap, so free them first

for (size\_t ix = 0; ix < overflow\_buffers\_ix; ix++) {

ei\_free(overflow\_buffers[ix]);

}

overflow\_buffers\_ix = 0;

return kTfLiteOk;

}

#os库

:mod:`os` -- basic "operating system" services

.. module:: os

:synopsis: basic "operating system" services

|see\_cpython\_module| :mod:`python:os`.

The ``os`` module contains functions for filesystem access and mounting,

terminal redirection and duplication, and the ``uname`` and ``urandom``

functions.

General functions

-----------------

.. function:: uname()

Return a tuple (possibly a named tuple) containing information about the

underlying machine and/or its operating system. The tuple has five fields

in the following order, each of them being a string:

\* ``sysname`` -- the name of the underlying system

\* ``nodename`` -- the network name (can be the same as ``sysname``)

\* ``release`` -- the version of the underlying system

\* ``version`` -- the MicroPython version and build date

\* ``machine`` -- an identifier for the underlying hardware (eg board, CPU)

.. function:: urandom(n)

Return a bytes object with \*n\* random bytes. Whenever possible, it is

generated by the hardware random number generator.

Filesystem access

-----------------

.. function:: chdir(path)

Change current directory.

.. function:: getcwd()

Get the current directory.

.. function:: ilistdir([dir])

This function returns an iterator which then yields tuples corresponding to

the entries in the directory that it is listing. With no argument it lists the

current directory, otherwise it lists the directory given by \*dir\*.

The tuples have the form \*(name, type, inode[, size])\*:

- \*name\* is a string (or bytes if \*dir\* is a bytes object) and is the name of

the entry;

- \*type\* is an integer that specifies the type of the entry, with 0x4000 for

directories and 0x8000 for regular files;

- \*inode\* is an integer corresponding to the inode of the file, and may be 0

for filesystems that don't have such a notion.

- Some platforms may return a 4-tuple that includes the entry's \*size\*. For

file entries, \*size\* is an integer representing the size of the file

or -1 if unknown. Its meaning is currently undefined for directory

entries.

.. function:: listdir([dir])

With no argument, list the current directory. Otherwise list the given directory.

.. function:: mkdir(path)

Create a new directory.

.. function:: remove(path)

Remove a file.

.. function:: rmdir(path)

Remove a directory.

.. function:: rename(old\_path, new\_path)

Rename a file.

.. function:: stat(path)

Get the status of a file or directory.

.. function:: statvfs(path)

Get the status of a filesystem.

Returns a tuple with the filesystem information in the following order:

\* ``f\_bsize`` -- file system block size

\* ``f\_frsize`` -- fragment size

\* ``f\_blocks`` -- size of fs in f\_frsize units

\* ``f\_bfree`` -- number of free blocks

\* ``f\_bavail`` -- number of free blocks for unprivileged users

\* ``f\_files`` -- number of inodes

\* ``f\_ffree`` -- number of free inodes

\* ``f\_favail`` -- number of free inodes for unprivileged users

\* ``f\_flag`` -- mount flags

\* ``f\_namemax`` -- maximum filename length

Parameters related to inodes: ``f\_files``, ``f\_ffree``, ``f\_avail``

and the ``f\_flags`` parameter may return ``0`` as they can be unavailable

in a port-specific implementation.

.. function:: sync()

Sync all filesystems.

Terminal redirection and duplication

------------------------------------

.. function:: dupterm(stream\_object, index=0, /)

Duplicate or switch the MicroPython terminal (the REPL) on the given :std:term:`stream`-like

object. The \*stream\_object\* argument must be a native stream object, or derive

from ``io.IOBase`` and implement the ``readinto()`` and

``write()`` methods. The stream should be in non-blocking mode and

``readinto()`` should return ``None`` if there is no data available for reading.

After calling this function all terminal output is repeated on this stream,

and any input that is available on the stream is passed on to the terminal input.

The \*index\* parameter should be a non-negative integer and specifies which

duplication slot is set. A given port may implement more than one slot (slot 0

will always be available) and in that case terminal input and output is

duplicated on all the slots that are set.

If ``None`` is passed as the \*stream\_object\* then duplication is cancelled on

the slot given by \*index\*.

The function returns the previous stream-like object in the given slot.

Filesystem mounting

-------------------

Some ports provide a Virtual Filesystem (VFS) and the ability to mount multiple

"real" filesystems within this VFS. Filesystem objects can be mounted at either

the root of the VFS, or at a subdirectory that lives in the root. This allows

dynamic and flexible configuration of the filesystem that is seen by Python

programs. Ports that have this functionality provide the :func:`mount` and

:func:`umount` functions, and possibly various filesystem implementations

represented by VFS classes.

.. function:: mount(fsobj, mount\_point, \*, readonly)

Mount the filesystem object \*fsobj\* at the location in the VFS given by the

\*mount\_point\* string. \*fsobj\* can be a a VFS object that has a ``mount()``

method, or a block device. If it's a block device then the filesystem type

is automatically detected (an exception is raised if no filesystem was

recognised). \*mount\_point\* may be ``'/'`` to mount \*fsobj\* at the root,

or ``'/<name>'`` to mount it at a subdirectory under the root.

If \*readonly\* is ``True`` then the filesystem is mounted read-only.

During the mount process the method ``mount()`` is called on the filesystem

object.

.. data:: IOCTL\_SET\_READOUT\_WINDOW

Lets you set the readout window for the OV5640.

.. data:: IOCTL\_GET\_READOUT\_WINDOW

Lets you get the readout window for the OV5640.

.. data:: IOCTL\_SET\_TRIGGERED\_MODE

Lets you set the triggered mode for the MT9V034.

.. data:: IOCTL\_GET\_TRIGGERED\_MODE

Lets you get the triggered mode for the MT9V034.

.. data:: IOCTL\_SET\_FOV\_WIDE

Enable `sensor.set\_framesize()` to optimize for the field-of-view over FPS.

.. data:: IOCTL\_GET\_FOV\_WIDE

Return if `sensor.set\_framesize()` is optimizing for field-of-view over FPS.

.. data:: IOCTL\_TRIGGER\_AUTO\_FOCUS

Used to trigger auto focus for the OV5640 FPC camera module.

.. data:: IOCTL\_PAUSE\_AUTO\_FOCUS

Used to pause auto focus (while running) for the OV5640 FPC camera module.

.. data:: IOCTL\_RESET\_AUTO\_FOCUS

Used to reset auto focus back to the default for the OV5640 FPC camera module.

.. data:: IOCTL\_WAIT\_ON\_AUTO\_FOCUS

Used to wait on auto focus to finish after being triggered for the OV5640 FPC camera module.

.. data:: IOCTL\_SET\_NIGHT\_MODE

Used to turn night mode on or off on a sensor. Nightmode reduces the frame rate to increase exposure dynamically.

.. data:: IOCTL\_GET\_NIGHT\_MODE

Gets the current value of if night mode is enabled or disabled for your sensor.

.. data:: IOCTL\_LEPTON\_GET\_WIDTH

Lets you get the FLIR Lepton image resolution width in pixels.

.. data:: IOCTL\_LEPTON\_GET\_HEIGHT

Lets you get the FLIR Lepton image resolution height in pixels.

.. data:: IOCTL\_LEPTON\_GET\_RADIOMETRY

Lets you get the FLIR Lepton type (radiometric or not).

.. data:: IOCTL\_LEPTON\_GET\_REFRESH

Lets you get the FLIR Lepton refresh rate in hertz.

.. data:: IOCTL\_LEPTON\_GET\_RESOLUTION

Lets you get the FLIR Lepton ADC resolution in bits.

.. data:: IOCTL\_LEPTON\_RUN\_COMMAND

Executes a 16-bit command given the FLIR Lepton SDK.

.. data:: IOCTL\_LEPTON\_SET\_ATTRIBUTE

Sets a FLIR Lepton Attribute given the FLIR Lepton SDK.

.. data:: IOCTL\_LEPTON\_GET\_ATTRIBUTE

Gets a FLIR Lepton Attribute given the FLIR Lepton SDK.

.. data:: IOCTL\_LEPTON\_GET\_FPA\_TEMPERATURE

Gets the FLIR Lepton FPA temp in celsius.

.. data:: IOCTL\_LEPTON\_GET\_AUX\_TEMPERATURE

Gets the FLIR Lepton AUX temp in celsius.

Will raise ``OSError(EPERM)`` if \*mount\_point\* is already mounted.

.. function:: umount(mount\_point)

Unmount a filesystem. \*mount\_point\* can be a string naming the mount location,

or a previously-mounted filesystem object. During the unmount process the

method ``umount()`` is called on the filesystem object.

Will raise ``OSError(EINVAL)`` if \*mount\_point\* is not found.

.. class:: VfsFat(block\_dev)

Create a filesystem object that uses the FAT filesystem format. Storage of

the FAT filesystem is provided by \*block\_dev\*.

Objects created by this constructor can be mounted using :func:`mount`.

.. staticmethod:: mkfs(block\_dev)

Build a FAT filesystem on \*block\_dev\*.

.. class:: VfsLfs1(block\_dev, readsize=32, progsize=32, lookahead=32)

Create a filesystem object that uses the `littlefs v1 filesystem format`\_.

Storage of the littlefs filesystem is provided by \*block\_dev\*, which must

support the :ref:`extended interface <block-device-interface>`.

Objects created by this constructor can be mounted using :func:`mount`.

See :ref:`filesystem` for more information.

.. staticmethod:: mkfs(block\_dev, readsize=32, progsize=32, lookahead=32)

Build a Lfs1 filesystem on \*block\_dev\*.

.. note:: There are reports of littlefs v1 failing in certain situations,

for details see `littlefs issue 347`\_.

.. class:: VfsLfs2(block\_dev, readsize=32, progsize=32, lookahead=32, mtime=True)

Create a filesystem object that uses the `littlefs v2 filesystem format`\_.

Storage of the littlefs filesystem is provided by \*block\_dev\*, which must

support the :ref:`extended interface <block-device-interface>`.

Objects created by this constructor can be mounted using :func:`mount`.

The \*mtime\* argument enables modification timestamps for files, stored using

littlefs attributes. This option can be disabled or enabled differently each

mount time and timestamps will only be added or updated if \*mtime\* is enabled,

otherwise the timestamps will remain untouched. Littlefs v2 filesystems without

timestamps will work without reformatting and timestamps will be added

transparently to existing files once they are opened for writing. When \*mtime\*

is enabled `os.stat` on files without timestamps will return 0 for the timestamp.

See :ref:`filesystem` for more information.

.. staticmethod:: mkfs(block\_dev, readsize=32, progsize=32, lookahead=32)

Build a Lfs2 filesystem on \*block\_dev\*.

.. note:: There are reports of littlefs v2 failing in certain situations,

for details see `littlefs issue 295`\_.

.. \_littlefs v1 filesystem format: https://github.com/ARMmbed/littlefs/tree/v1

.. \_littlefs v2 filesystem format: https://github.com/ARMmbed/littlefs

.. \_littlefs issue 295: https://github.com/ARMmbed/littlefs/issues/295

.. \_littlefs issue 347: https://github.com/ARMmbed/littlefs/issues/347

Block devices

-------------

A block device is an object which implements the block protocol. This enables a

device to support MicroPython filesystems. The physical hardware is represented

by a user defined class. The :class:`AbstractBlockDev` class is a template for

the design of such a class: MicroPython does not actually provide that class,

but an actual block device class must implement the methods described below.

A concrete implementation of this class will usually allow access to the

memory-like functionality of a piece of hardware (like flash memory). A block

device can be formatted to any supported filesystem and mounted using ``os``

methods.

See :ref:`filesystem` for example implementations of block devices using the

two variants of the block protocol described below.

.. \_block-device-interface:

Simple and extended interface

.............................

There are two compatible signatures for the ``readblocks`` and ``writeblocks``

methods (see below), in order to support a variety of use cases. A given block

device may implement one form or the other, or both at the same time. The second

form (with the offset parameter) is referred to as the "extended interface".

Some filesystems (such as littlefs) that require more control over write

operations, for example writing to sub-block regions without erasing, may require

that the block device supports the extended interface.

.. class:: AbstractBlockDev(...)

Construct a block device object. The parameters to the constructor are

dependent on the specific block device.

.. method:: readblocks(block\_num, buf)

readblocks(block\_num, buf, offset)

The first form reads aligned, multiples of blocks.

Starting at the block given by the index \*block\_num\*, read blocks from

the device into \*buf\* (an array of bytes).

The number of blocks to read is given by the length of \*buf\*,

which will be a multiple of the block size.

The second form allows reading at arbitrary locations within a block,

and arbitrary lengths.

Starting at block index \*block\_num\*, and byte offset within that block

of \*offset\*, read bytes from the device into \*buf\* (an array of bytes).

The number of bytes to read is given by the length of \*buf\*.

.. method:: writeblocks(block\_num, buf)

writeblocks(block\_num, buf, offset)

The first form writes aligned, multiples of blocks, and requires that the

blocks that are written to be first erased (if necessary) by this method.

Starting at the block given by the index \*block\_num\*, write blocks from

\*buf\* (an array of bytes) to the device.

The number of blocks to write is given by the length of \*buf\*,

which will be a multiple of the block size.

The second form allows writing at arbitrary locations within a block,

and arbitrary lengths. Only the bytes being written should be changed,

and the caller of this method must ensure that the relevant blocks are

erased via a prior ``ioctl`` call.

Starting at block index \*block\_num\*, and byte offset within that block

of \*offset\*, write bytes from \*buf\* (an array of bytes) to the device.

The number of bytes to write is given by the length of \*buf\*.

Note that implementations must never implicitly erase blocks if the offset

argument is specified, even if it is zero.

.. method:: ioctl(op, arg)

Control the block device and query its parameters. The operation to

perform is given by \*op\* which is one of the following integers:

- 1 -- initialise the device (\*arg\* is unused)

- 2 -- shutdown the device (\*arg\* is unused)

- 3 -- sync the device (\*arg\* is unused)

- 4 -- get a count of the number of blocks, should return an integer

(\*arg\* is unused)

- 5 -- get the number of bytes in a block, should return an integer,

or ``None`` in which case the default value of 512 is used

(\*arg\* is unused)

- 6 -- erase a block, \*arg\* is the block number to erase

As a minimum ``ioctl(4, ...)`` must be intercepted; for littlefs

``ioctl(6, ...)`` must also be intercepted. The need for others is

hardware dependent.

Prior to any call to ``writeblocks(block, ...)`` littlefs issues

``ioctl(6, block)``. This enables a device driver to erase the block

prior to a write if the hardware requires it. Alternatively a driver

might intercept ``ioctl(6, block)`` and return 0 (success). In this case

the driver assumes responsibility for detecting the need for erasure.

Unless otherwise stated ``ioctl(op, arg)`` can return ``None``.

Consequently an implementation can ignore unused values of ``op``. Where

``op`` is intercepted, the return value for operations 4 and 5 are as

detailed above. Other operations should return 0 on success and non-zero

for failure, with the value returned being an ``OSError`` errno code.

#image库

:mod:`sensor` --- camera sensor

===============================

.. module:: sensor

:synopsis: camera sensor

The ``sensor`` module is used for taking pictures.

Example usage::

import sensor

# Setup camera.

sensor.reset()

sensor.set\_pixformat(sensor.RGB565)

sensor.set\_framesize(sensor.QVGA)

sensor.skip\_frames()

# Take pictures.

while(True):

sensor.snapshot()

Functions

---------

.. function:: reset()

Initializes the camera sensor.

.. function:: sleep(enable)

Puts the camera to sleep if enable is True. Otherwise, wakes it back up.

.. function:: shutdown(enable)

Puts the camera into a lower power mode than sleep (but the camera must be reset on being woken up).

.. function:: flush()

Copies whatever was in the frame buffer to the IDE. You should call this

method to display the last image your OpenMV Cam takes if it's not running

a script with an infinite loop. Note that you'll need to add a delay time

of about a second after your script finishes for the IDE to grab the image

from your camera. Otherwise, this method will have no effect.

.. function:: snapshot()

Takes a picture using the camera and returns an ``image`` object.

The OpenMV Cam has two memory areas for images. The classical stack/heap

area used for normal MicroPython processing can store small images within

it's heap. However, the MicroPython heap is only about ~100 KB which is not

enough to store larger images. So, your OpenMV Cam has a secondary frame

buffer memory area that stores images taken by `sensor.snapshot()`. Images

are stored on the bottom of this memory area. Any memory that's left

over is then available for use by the frame buffer stack which your OpenMV

Cam's firmware uses to hold large temporary data structures for image

processing algorithms.

If you need room to hold multiple frames you may "steal" frame buffer space

by calling `sensor.alloc\_extra\_fb()`.

If `sensor.set\_auto\_rotation()` is enabled this method will return a new

already rotated `image` object.

.. note::

`sensor.snapshot()` may apply cropping parameters to fit the snapshot in the available

RAM the pixformat, framesize, windowing, and framebuffers. The cropping parameters will be applied

to maintain the aspect ratio and will stay until `sensor.set\_framesize()` or `sensor.set\_windowing()` are called.

.. function:: skip\_frames([n, time])

Takes ``n`` number of snapshots to let the camera image stabilize after

changing camera settings. ``n`` is passed as normal argument, e.g.

``skip\_frames(10)`` to skip 10 frames. You should call this function after

changing camera settings.

Alternatively, you can pass the keyword argument ``time`` to skip frames

for some number of milliseconds, e.g. ``skip\_frames(time = 2000)`` to skip

frames for 2000 milliseconds.

If neither ``n`` nor ``time`` is specified this method skips frames for

300 milliseconds.

If both are specified this method skips ``n`` number of frames but will

timeout after ``time`` milliseconds.

.. note::

`sensor.snapshot()` may apply cropping parameters to fit the snapshot in the available

RAM given the pixformat, framesize, windowing, and framebuffers. The cropping parameters will be applied

to maintain the aspect ratio and will stay until `sensor.set\_framesize()` or `sensor.set\_windowing()` are called.

.. function:: width()

Returns the sensor resolution width.

.. function:: height()

Returns the sensor resolution height.

.. function:: get\_fb()

(Get Frame Buffer) Returns the image object returned by a previous call of

`sensor.snapshot()`. If `sensor.snapshot()` had not been called before

then ``None`` is returned.

.. function:: get\_id()

Returns the camera module ID.

\* `sensor.OV9650`: First gen OpenMV Cam sensor - never released.

\* `sensor.OV2640`: Second gen OpenMV Cam sensor - never released.

\* `sensor.OV5640`: High-res OpenMV Cam H7 sensor.

\* `sensor.OV7725`: Rolling shutter sensor module.

\* `sensor.OV7690`: OpenMV Cam Micro sensor module.

\* `sensor.MT9V034`: Global shutter sensor module.

\* `sensor.MT9M114`: New Rolling shutter sensor module.

\* `sensor.LEPTON`: Lepton1/2/3 sensor module.

\* `sensor.HM01B0`: Arduino Portenta H7 sensor module.

\* `sensor.GC2145`: Arduino Nicla Vision H7 sensor module.

\* `sensor.PAJ6100`: PixArt Imaging sensor Module.

.. function:: alloc\_extra\_fb(width, height, pixformat)

Allocates another frame buffer for image storage from the frame buffer stack

and returns an ``image`` object of ``width``, ``height``, and ``pixformat``.

You may call this function as many times as you like as long as there's

memory available to allocate any number of extra frame buffers.

\*wbits\* allows you to configure the DEFLATE dictionary window size and the

output format. The window size allows you to trade-off memory usage for

compression level. A larger window size will allow the compressor to

reference fragments further back in the input. The output formats are "raw"

DEFLATE (no header/footer), zlib, and gzip, where the latter two

include a header and checksum.

The low four bits of the absolute value of \*wbits\* set the base-2 logarithm of

the DEFLATE dictionary window size. So for example, ``wbits=10``,

``wbits=-10``, and ``wbits=26`` all set the window size to 1024 bytes. Valid

window sizes are ``5`` to ``15`` inclusive (corresponding to 32 to 32k bytes).

Negative values of \*wbits\* between ``-5`` and ``-15`` correspond to "raw"

output mode, positive values between ``5`` and ``15`` correspond to zlib

output mode, and positive values between ``21`` and ``31`` correspond to

gzip output mode.

See the :mod:`CPython documentation for zlib <python:zlib>` for more

information about the \*wbits\* parameter. Note that MicroPython allows

for smaller window sizes, which is useful when memory is constrained while

still achieving a reasonable level of compression. It also speeds up

the compressor. See more :ref:`MicroPython-specific details <deflate\_wbits>`

in the :mod:`deflate <deflate>` module documentation.

If ``pixformat`` is a number >= 4 then this will allocate a JPEG image. You

can then do `Image.bytearray()` to get byte level read/write access to the JPEG image.

.. note::

Creating secondary images normally requires creating them on the heap which

has a limited amount of RAM... but, also gets fragmented making it hard to

grab a large contigous memory array to store an image in. With this method

you are able to allocate a very large memory array for an image instantly

by taking space away from our frame buffer stack memory which we use for

computer vision algorithms. That said, this also means you'll run out of

memory more easily if you try to execute more memory intensive machine

vision algorithms like `Image.find\_apriltags()`.

.. function:: dealloc\_extra\_fb()

Deallocates the last previously allocated extra frame buffer. Extra frame

buffers are stored in a stack like structure.

.. note::

Your OpenMV Cam has two memory areas. First, you have your classical

.data/.bss/heap/stack memory area. The .data/.bss/heap regions are

fixed by firmware. The stack then grows down until it hits the heap.

Next, frame buffers are stored in a secondary memory region. Memory is

liad out with the main frame buffer on the bottom and the frame buffer

stack on the top. When `sensor.snapshot()` is called it fills the frame bufer

from the bottom. The frame buffer stack is then able to use whatever is

left over. This memory allocation method is extremely efficent for computer

vision on microcontrollers.

.. function:: set\_pixformat(pixformat)

Sets the pixel format for the camera module.

\* `sensor.GRAYSCALE`: 8-bits per pixel.

\* `sensor.RGB565`: 16-bits per pixel.

\* `sensor.BAYER`: 8-bits per pixel bayer pattern.

\* `sensor.YUV422`: 16-bits per pixel (8-bits Y1, 8-bits U, 8-bits Y2, 8-bits V, etc.)

\* `sensor.JPEG`: Compressed JPEG data. Only for the OV2640/OV5640.

If you are trying to take JPEG images with the OV2640 or OV5640 camera modules at high

resolutions you should set the pixformat to `sensor.JPEG`. You can control the image

quality then with `sensor.set\_quality()`.

.. function:: get\_pixformat()

Returns the pixformat for the camera module.

.. function:: set\_framesize(framesize)

Sets the frame size for the camera module.

\* `sensor.QQCIF`: 88x72

\* `sensor.QCIF`: 176x144

\* `sensor.CIF`: 352x288

\* `sensor.QQSIF`: 88x60

\* `sensor.QSIF`: 176x120

\* `sensor.SIF`: 352x240

\* `sensor.QQQQVGA`: 40x30

\* `sensor.QQQVGA`: 80x60

\* `sensor.QQVGA`: 160x120

\* `sensor.QVGA`: 320x240

\* `sensor.VGA`: 640x480

\* `sensor.HQQQQVGA`: 30x20

\* `sensor.HQQQVGA`: 60x40

\* `sensor.HQQVGA`: 120x80

\* `sensor.HQVGA`: 240x160

\* `sensor.HVGA`: 480x320

\* `sensor.B64X32`: 64x32 (for use with `Image.find\_displacement()`)

\* `sensor.B64X64`: 64x64 (for use with `Image.find\_displacement()`)

\* `sensor.B128X64`: 128x64 (for use with `Image.find\_displacement()`)

\* `sensor.B128X128`: 128x128 (for use with `Image.find\_displacement()`)

\* `sensor.B160X160`: 160x160 (for the HM01B0)

\* `sensor.B320X320`: 320x320 (for the HM01B0)

\* `sensor.LCD`: 128x160 (for use with the lcd shield)

\* `sensor.QQVGA2`: 128x160 (for use with the lcd shield)

\* `sensor.WVGA`: 720x480 (for the MT9V034)

\* `sensor.WVGA2`:752x480 (for the MT9V034)

\* `sensor.SVGA`: 800x600 (only for the OV2640/OV5640 sensor)

\* `sensor.XGA`: 1024x768 (only for the OV2640/OV5640 sensor)

\* `sensor.WXGA`: 1280x768 (for the MT9M114)

\* `sensor.SXGA`: 1280x1024 (only for the OV2640/OV5640 sensor)

\* `sensor.SXGAM`: 1280x960 (for the MT9M114)

\* `sensor.UXGA`: 1600x1200 (only for the OV2640/OV5640 sensor)

\* `sensor.HD`: 1280x720 (only for the OV2640/OV5640 sensor)

\* `sensor.FHD`: 1920x1080 (only for the OV5640 sensor)

\* `sensor.QHD`: 2560x1440 (only for the OV5640 sensor)

\* `sensor.QXGA`: 2048x1536 (only for the OV5640 sensor)

\* `sensor.WQXGA`: 2560x1600 (only for the OV5640 sensor)

\* `sensor.WQXGA2`: 2592x1944 (only for the OV5640 sensor)

.. function:: get\_framesize()

Returns the frame size for the camera module.

.. function:: set\_framerate(rate)

Sets the frame rate in hz for the camera module.

.. note::

`set\_framerate` works by dropping frames received by the camera module to keep the frame rate

equal to (or below) the rate you specify. By default the camera will run at the maximum frame

rate. If implemented for the particular camera sensor then `set\_framerate` will also reduce

the camera sensor frame rate internally to save power and improve image quality by increasing

the sensor exposure. `set\_framerate` may conflict with `set\_auto\_exposure` on some cameras.

.. function:: get\_framerate()

Returns the frame rate in hz for the camera module.

.. function:: set\_windowing(roi)

Sets the resolution of the camera to a sub resolution inside of the current

resolution. For example, setting the resolution to `sensor.VGA` and then

the windowing to (120, 140, 200, 200) sets `sensor.snapshot()` to capture

the 200x200 center pixels of the VGA resolution outputted by the camera

sensor. You can use windowing to get custom resolutions. Also, when using

windowing on a larger resolution you effectively are digital zooming.

``roi`` is a rect tuple (x, y, w, h). However, you may just pass (w, h) and

the ``roi`` will be centered on the frame. You may also pass roi not in parens.

This function will automatically handle cropping the passed roi to the framesize.

.. function:: get\_windowing()

Returns the ``roi`` tuple (x, y, w, h) previously set with `sensor.set\_windowing()`.

.. function:: set\_gainceiling(gainceiling)

Set the camera image gainceiling. 2, 4, 8, 16, 32, 64, or 128.

.. function:: set\_contrast(constrast)

Set the camera image contrast. -3 to +3.

.. function:: set\_brightness(brightness)

Set the camera image brightness. -3 to +3.

.. function:: set\_saturation(saturation)

Set the camera image saturation. -3 to +3.

.. function:: set\_quality(quality)

Set the camera image JPEG compression quality. 0 - 100.

.. note::

Only for the OV2640/OV5640 cameras.

.. function:: set\_colorbar(enable)

Turns color bar mode on (True) or off (False). Defaults to off.

.. function:: set\_auto\_gain(enable, [gain\_db=-1, [gain\_db\_ceiling]])

``enable`` turns auto gain control on (True) or off (False).

The camera will startup with auto gain control on.

If ``enable`` is False you may set a fixed gain in decibels with ``gain\_db``.

If ``enable`` is True you may set the maximum gain ceiling in decibels with

``gain\_db\_ceiling`` for the automatic gain control algorithm.

.. note::

You need to turn off white balance too if you want to track colors.

.. function:: get\_gain\_db()

Returns the current camera gain value in decibels (float).

.. function:: set\_auto\_exposure(enable, [exposure\_us])

``enable`` turns auto exposure control on (True) or off (False).

The camera will startup with auto exposure control on.

If ``enable`` is False you may set a fixed exposure time in microseconds

with ``exposure\_us``.

.. note::

Camera auto exposure algorithms are pretty conservative about how much

they adjust the exposure value by and will generally avoid changing the

exposure value by much. Instead, they change the gain value alot of deal

with changing lighting.

.. function:: get\_exposure\_us()

Returns the current camera exposure value in microseconds (int).

.. function:: set\_auto\_whitebal(enable, [rgb\_gain\_db])

``enable`` turns auto white balance on (True) or off (False).

The camera will startup with auto white balance on.

If ``enable`` is False you may set a fixed gain in decibels for the red, green,

and blue channels respectively with ``rgb\_gain\_db``.

.. note::

You need to turn off gain control too if you want to track colors.

.. function:: get\_rgb\_gain\_db()

Returns a tuple with the current camera red, green, and blue gain values in

decibels ((float, float, float)).

.. function:: set\_auto\_blc([enable, [regs]])

Sets the auto black line calibration (blc) control on the camera.

``enable`` pass `True` or `False` to turn BLC on or off. You typically always want this on.

``regs`` if disabled then you can manually set the blc register values via the values you

got previously from `get\_blc\_regs()`.

.. function:: get\_blc\_regs()

Returns the sensor blc registers as an opaque tuple of integers. For use with `set\_auto\_blc`.

.. function:: set\_hmirror(enable)

Turns horizontal mirror mode on (True) or off (False). Defaults to off.

.. function:: get\_hmirror()

Returns if horizontal mirror mode is enabled.

.. function:: set\_vflip(enable)

Turns vertical flip mode on (True) or off (False). Defaults to off.

.. function:: get\_vflip()

Returns if vertical flip mode is enabled.

.. function:: set\_transpose(enable)

Turns transpose mode on (True) or off (False). Defaults to off.

\* vflip=False, hmirror=False, transpose=False -> 0 degree rotation

\* vflip=True, hmirror=False, transpose=True -> 90 degree rotation

\* vflip=True, hmirror=True, transpose=False -> 180 degree rotation

\* vflip=False, hmirror=True, transpose=True -> 270 degree rotation

.. function:: get\_transpose()

Returns if transpose mode is enabled.

.. function:: set\_auto\_rotation(enable)

Turns auto rotation mode on (True) or off (False). Defaults to off.

.. note::

This function only works when the OpenMV Cam has an `imu` installed and is enabled automatically.

.. function:: get\_auto\_rotation()

Returns if auto rotation mode is enabled.

.. note::

This function only works when the OpenMV Cam has an `imu` installed and is enabled automatically.

.. function:: set\_framebuffers(count)

Sets the number of frame buffers used to receive image data. By default your OpenMV Cam will

automatically try to allocate the maximum number of frame buffers it can possibly allocate

without using more than 1/2 of the available frame buffer RAM at the time of allocation to

ensure the best performance. Automatic reallocation of frame buffers occurs whenever you

call `sensor.set\_pixformat()`, `sensor.set\_framesize()`, and `sensor.set\_windowing()`.

`sensor.snapshot()` will automatically handle switching active frame buffers in the background.

From your code's perspective there is only ever 1 active frame buffer even though there might

be more than 1 frame buffer on the system and another frame buffer reciving data in the background.

If count is:

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*1\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

1 - Single Buffer Mode (you may also pass `sensor.SINGLE\_BUFFER`)

In single buffer mode your OpenMV Cam will allocate one frame buffer for receiving images.

When you call `sensor.snapshot()` that framebuffer will be used to receive the image and

the camera driver will continue to run. In the advent you call `sensor.snapshot()` again

before the first line of the next frame is received your code will execute at the frame rate

of the camera. Otherwise, the image will be dropped.

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*2\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

2 - Double Buffer Mode (you may also pass `sensor.DOUBLE\_BUFFER`)

In double buffer mode your OpenMV Cam will allocate two frame buffers for receiving images.

When you call `sensor.snapshot()` one framebuffer will be used to receive the image and

the camera driver will continue to run. When the next frame is received it will be stored

in the other frame bufer. In the advent you call `sensor.snapshot()` again

before the first line of the next frame after is received your code will execute at the frame rate

of the camera. Otherwise, the image will be dropped.

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*3\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

3 - Triple Buffer Mode (you may also pass `sensor.TRIPLE\_BUFFER`)

In triple buffer mode your OpenMV Cam will allocate three buffers for receiving images.

In this mode there is always a frame buffer to store the received image to in the background

resulting in the highest performance and lowest latency for reading the latest received frame.

No frames are ever dropped in this mode. The next frame read by `sensor.snapshot()` is the

last captured frame by the sensor driver (e.g. if you are reading slower than the camera

frame rate then the older frame in the possible frames available is skipped).

Regarding the reallocation above, triple buffering is tried first, then double buffering, and if

these both fail to fit in 1/2 of the available frame buffer RAM then single buffer mode is used.

You may pass a value of 4 or greater to put the sensor driver into video FIFO mode where received

images are stored in a frame buffer FIFO with ``count`` buffers. This is useful for video recording

to an SD card which may randomly block your code from writing data when the SD card is performing

house-keeping tasks like pre-erasing blocks to write data to.

.. note::

On frame drop (no buffers available to receive the next frame) all frame buffers are automatically

cleared except the active frame buffer. This is done to ensure `sensor.snapshot()` returns current

frames and not frames from long ago.

Fun fact, you can pass a value of 100 or so on OpenMV Cam's with SDRAM for a huge video fifo. If

you then call snapshot slower than the camera frame rate (by adding `machine.sleep()`) you'll get

slow-mo effects in OpenMV IDE. However, you will also see the above policy effect of resetting

the frame buffer on a frame drop to ensure that frames do not get too old. If you want to record

slow-mo video just record video normally to the SD card and then play the video back on a desktop

machine slower than it was recorded.

.. function:: get\_framebuffers()

Returns the current number of frame buffers allocated.

.. function:: disable\_delays([disable])

If ``disable`` is ``True`` then disable all settling time delays in the sensor module.

Whenever you reset the camera module, change modes, etc. the sensor driver delays to prevent

you can from calling `snapshot` to quickly afterwards and receiving corrupt frames from the

camera module. By disabling delays you can quickly update the camera module settings in bulk

via multiple function calls before delaying at the end and calling `snapshot`.

If this function is called with no arguments it returns if delays are disabled.

.. function:: disable\_full\_flush([disable])

If ``disable`` is ``True`` then automatic framebuffer flushing mentioned in `set\_framebuffers`

is disabled. This removes any time limit on frames in the frame buffer fifo. For example, if

you set the number of frame buffers to 30 and set the frame rate to 30 you can now precisely

record 1 second of video from the camera without risk of frame loss.

If this function is called with no arguments it returns if automatic flushing is disabled. By

default automatic flushing on frame drop is enabled to clear out stale frames.

.. note::

`snapshot` starts the frame capture process which will continue to capture frames until

there is no space to hold a frame at which point the frame capture process stops. The

process always stops when there is no space to hold the next frame.

.. function:: set\_lens\_correction(enable, radi, coef)

``enable`` True to enable and False to disable (bool).

``radi`` integer radius of pixels to correct (int).

``coef`` power of correction (int).

.. function:: set\_vsync\_callback(cb)

Registers callback ``cb`` to be executed (in interrupt context) whenever the camera module

generates a new frame (but, before the frame is received).

``cb`` takes one argument and is passed the current state of the vsync pin after changing.

.. function:: set\_frame\_callback(cb)

Registers callback ``cb`` to be executed (in interrupt context) whenever the camera module

generates a new frame and the frame is ready to be read via `sensor.snapshot()`.

``cb`` takes no arguments.

Use this to get an interrupt to schedule reading a frame later with `micropython.schedule()`.

.. function:: get\_frame\_available()

Returns True if a frame is available to read by calling `sensor.snapshot()`.

.. function:: ioctl(...)

Executes a sensor specific method:

\* `sensor.IOCTL\_SET\_READOUT\_WINDOW` - Pass this enum followed by a rect tuple (x, y, w, h) or a size tuple (w, h).

\* This IOCTL allows you to control the readout window of the camera sensor which dramatically improves the frame rate at the cost of field-of-view.

\* If you pass a rect tuple (x, y, w, h) the readout window will be positoned on that rect tuple. The rect tuple's x/y position will be adjusted so the size w/h fits. Additionally, the size w/h will be adjusted to not be smaller than the ``framesize``.

\* If you pass a size tuple (w, h) the readout window will be centered given the w/h. Additionally, the size w/h will be adjusted to not be smaller than the ``framesize``.

\* This IOCTL is extremely helpful for increasing the frame rate on higher resolution cameras like the OV2640/OV5640.

\* `sensor.IOCTL\_GET\_READOUT\_WINDOW` - Pass this enum for `sensor.ioctl` to return the current readout window rect tuple (x, y, w, h). By default this is (0, 0, maximum\_camera\_sensor\_pixel\_width, maximum\_camera\_sensor\_pixel\_height).

\* `sensor.IOCTL\_SET\_TRIGGERED\_MODE` - Pass this enum followed by True or False set triggered mode for the MT9V034 sensor.

\* `sensor.IOCTL\_GET\_TRIGGERED\_MODE` - Pass this enum for `sensor.ioctl` to return the current triggered mode state.

\* `sensor.IOCTL\_SET\_FOV\_WIDE` - Pass this enum followed by True or False enable `sensor.set\_framesize()` to optimize for the field-of-view over FPS.

\* `sensor.IOCTL\_GET\_FOV\_WIDE` - Pass this enum for `sensor.ioctl` to return the current field-of-view over fps optimization state.

\* `sensor.IOCTL\_TRIGGER\_AUTO\_FOCUS` - Pass this enum for `sensor.ioctl` to trigger auto focus on the OV5640 FPC camera module.

\* `sensor.IOCTL\_PAUSE\_AUTO\_FOCUS` - Pass this enum for `sensor.ioctl` to pause auto focus (after triggering) on the OV5640 FPC camera module.

\* `sensor.IOCTL\_RESET\_AUTO\_FOCUS` - Pass this enum for `sensor.ioctl` to reset auto focus (after triggering) on the OV5640 FPC camera module.

\* `sensor.IOCTL\_WAIT\_ON\_AUTO\_FOCUS` - Pass this enum for `sensor.ioctl` to wait for auto focus (after triggering) to finish on the OV5640 FPC camera module. You may pass a second argument of the timeout in milliseconds. The default is 5000 ms.

\* `sensor.IOCTL\_SET\_NIGHT\_MODE` - Pass this enum followed by True or False set nightmode the OV7725 and OV5640 sensors.

\* `sensor.IOCTL\_GET\_NIGHT\_MODE` - Pass this enum for `sensor.ioctl` to return the current night mode state.

\* `sensor.IOCTL\_LEPTON\_GET\_WIDTH` - Pass this enum to get the FLIR Lepton image width in pixels.

\* `sensor.IOCTL\_LEPTON\_GET\_HEIGHT` - Pass this enum to get the FLIR Lepton image height in pixels.

\* `sensor.IOCTL\_LEPTON\_GET\_RADIOMETRY` - Pass this enum to get the FLIR Lepton type (radiometric or not).

\* `sensor.IOCTL\_LEPTON\_GET\_REFRESH` - Pass this enum to get the FLIR Lepton refresh rate in hertz.

\* `sensor.IOCTL\_LEPTON\_GET\_RESOLUTION` - Pass this enum to get the FLIR Lepton ADC resolution in bits.

\* `sensor.IOCTL\_LEPTON\_RUN\_COMMAND` - Pass this enum to execute a FLIR Lepton SDK command. You need to pass an additional 16-bit value after the enum as the command to execute.

\* `sensor.IOCTL\_LEPTON\_SET\_ATTRIBUTE` - Pass this enum to set a FLIR Lepton SDK attribute.

\* The first argument is the 16-bit attribute ID to set (set the FLIR Lepton SDK).

\* The second argument is a MicroPython byte array of bytes to write (should be a multiple of 16-bits). Create the byte array using ``struct`` following the FLIR Lepton SDK.

\* `sensor.IOCTL\_LEPTON\_GET\_ATTRIBUTE` - Pass this enum to get a FLIR Lepton SDK attribute.

\* The first argument is the 16-bit attribute ID to set (set the FLIR Lepton SDK).

\* Returns a MicroPython byte array of the attribute. Use ``struct`` to deserialize the byte array following the FLIR Lepton SDK.

\* `sensor.IOCTL\_LEPTON\_GET\_FPA\_TEMPERATURE` - Pass this enum to get the FLIR Lepton FPA Temp in celsius.

\* `sensor.IOCTL\_LEPTON\_GET\_AUX\_TEMPERATURE` - Pass this enum to get the FLIR Lepton AUX Temp in celsius.

\* `sensor.IOCTL\_LEPTON\_SET\_MEASUREMENT\_MODE` - Pass this followed by True or False to turn off automatic gain control on the FLIR Lepton and force it to output an image where each pixel value represents an exact temperature value in celsius. A second True enables high temperature mode enabling measurements up to 500C on the Lepton 3.5, False is the default low temperature mode.

\* `sensor.IOCTL\_LEPTON\_GET\_MEASUREMENT\_MODE` - Pass this to get a tuple for (measurement-mode-enabled, high-temp-enabled).

\* `sensor.IOCTL\_LEPTON\_SET\_MEASUREMENT\_RANGE` - Pass this when measurement mode is enabled to set the temperature range in celsius for the mapping operation. The temperature image returned by the FLIR Lepton will then be clamped between these min and max values and then scaled to values between 0 to 255. To map a pixel value back to a temperature (on a grayscale image) do: ((pixel \* (max\_temp\_in\_celsius - min\_temp\_in\_celsius)) / 255.0) + min\_temp\_in\_celsius.

\* The first arugment should be the min temperature in celsius.

\* The second argument should be the max temperature in celsius. If the arguments are reversed the library will automatically swap them for you.

\* `sensor.IOCTL\_LEPTON\_GET\_MEASUREMENT\_RANGE` - Pass this to return the sorted (min, max) 2 value temperature range tuple. The default is -10C to 40C if not set yet.

\* `sensor.IOCTL\_HIMAX\_MD\_ENABLE` - Pass this enum followed by ``True``/``False`` to enable/disable motion detection on the HM01B0. You should also enable the I/O pin (PC15 on the Arduino Portenta) attached the HM01B0 motion detection line to receive an interrupt.

\* `sensor.IOCTL\_HIMAX\_MD\_CLEAR` - Pass this enum to clear the motion detection interrupt on the HM01B0.

\* `sensor.IOCTL\_HIMAX\_MD\_WINDOW` - Pass this enum followed by (x1, y1, x2, y2) to set the motion detection window on the HM01B0.

\* `sensor.IOCTL\_HIMAX\_MD\_THRESHOLD` - Pass this enum followed by a threshold value (0-255) to set the motion detection threshold on the HM01B0.

\* `sensor.IOCTL\_HIMAX\_OSC\_ENABLE` - Pass this enum followed by ``True``/``False`` to enable/disable the oscillator HM01B0 to save power.

.. function:: set\_color\_palette(palette)

Sets the color palette to use for FLIR Lepton grayscale to RGB565 conversion.

.. function:: get\_color\_palette()

Returns the current color palette setting. Defaults to `image.PALETTE\_RAINBOW`.

.. function:: \_\_write\_reg(address, value)

Write ``value`` (int) to camera register at ``address`` (int).

.. note:: See the camera data sheet for register info.

.. function:: \_\_read\_reg(address)

Read camera register at ``address`` (int).

.. note:: See the camera data sheet for register info.

Constants

-------------------------------------------------------------------------------------------------------------------------------------------

.. data:: BINARY

BINARY (bitmap) pixel format. Each pixel is 1-bit.

This format is usful for mask storage. Can be used with `image.Image()` and

`sensor.alloc\_extra\_fb()`.

.. data:: GRAYSCALE

GRAYSCALE pixel format (Y from YUV422). Each pixel is 8-bits, 1-byte.

All of our computer vision algorithms run faster on grayscale images than

RGB565 images.

.. data:: RGB565

RGB565 pixel format. Each pixel is 16-bits, 2-bytes. 5-bits are used for red,

6-bits are used for green, and 5-bits are used for blue.

All of our computer vision algorithms run slower on RGB565 images than

grayscale images.

.. data:: BAYER

RAW BAYER image pixel format. If you try to make the frame size too big

to fit in the frame buffer your OpenMV Cam will set the pixel format

to BAYER so that you can capture images but only some image processing methods

will be operational.

.. data:: YUV422

A pixel format that is very easy to jpeg compress. Each pixel is stored as a grayscale

8-bit Y value followed by alternating 8-bit U/V color values that are shared between two

Y values (8-bits Y1, 8-bits U, 8-bits Y2, 8-bits V, etc.). Only some image processing

methods work with YUV422.

.. data:: JPEG

JPEG mode. The camera module outputs compressed jpeg images.

Use `sensor.set\_quality()` to control the jpeg quality.

Only works for the OV2640/OV5640 cameras.

.. data:: OV2640

`sensor.get\_id()` returns this for the OV2640 camera.

.. data:: OV5640

`sensor.get\_id()` returns this for the OV5640 camera.

.. data:: OV7690

`sensor.get\_id()` returns this for the OV7690 camera.

.. data:: OV7725

`sensor.get\_id()` returns this for the OV7725 camera.

.. data:: OV9650

`sensor.get\_id()` returns this for the OV9650 camera.

.. data:: MT9V022

`sensor.get\_id()` returns this for the MT9V022 camera.

.. data:: MT9V024

`sensor.get\_id()` returns this for the MT9V024 camera.

.. data:: MT9V032

`sensor.get\_id()` returns this for the MT9V032 camera.

.. data:: MT9V034

`sensor.get\_id()` returns this for the MT9V034 camera.

.. data:: MT9M114

`sensor.get\_id()` returns this for the MT9M114 camera.

.. data:: LEPTON

`sensor.get\_id()` returns this for the LEPTON1/2/3 cameras.

.. data:: HM01B0

`sensor.get\_id()` returns this for the HM01B0 camera.

.. data:: HM0360

`sensor.get\_id()` returns this for the HM01B0 camera.

.. data:: GC2145

`sensor.get\_id()` returns this for the GC2145 camera.

.. data:: PAJ6100

`sensor.get\_id()` returns this for the PAJ6100 camera.

.. data:: FROGEYE2020

`sensor.get\_id()` returns this for the FROGEYE2020 camera.

.. data:: QQCIF

88x72 resolution for the camera sensor.

.. data:: QCIF

176x144 resolution for the camera sensor.

.. data:: CIF

352x288 resolution for the camera sensor.

.. data:: QQSIF

88x60 resolution for the camera sensor.

.. data:: QSIF

176x120 resolution for the camera sensor.

.. data:: SIF

352x240 resolution for the camera sensor.

.. data:: QQQQVGA

40x30 resolution for the camera sensor.

.. data:: QQQVGA

80x60 resolution for the camera sensor.

.. data:: QQVGA

160x120 resolution for the camera sensor.

.. data:: QVGA

320x240 resolution for the camera sensor.

.. data:: VGA

640x480 resolution for the camera sensor.

.. data:: HQQQQVGA

30x20 resolution for the camera sensor.

.. data:: HQQQVGA

60x40 resolution for the camera sensor.

.. data:: HQQVGA

120x80 resolution for the camera sensor.

.. data:: HQVGA

240x160 resolution for the camera sensor.

.. data:: HVGA

480x320 resolution for the camera sensor.

.. data:: B64X32

64x32 resolution for the camera sensor.

For use with `Image.find\_displacement()` and any other FFT based algorithm.

.. data:: B64X64

64x64 resolution for the camera sensor.

For use with `Image.find\_displacement()` and any other FFT based algorithm.

.. data:: B128X64

128x64 resolution for the camera sensor.

For use with `Image.find\_displacement()` and any other FFT based algorithm.

.. data:: B128X128

128x128 resolution for the camera sensor.

For use with `Image.find\_displacement()` and any other FFT based algorithm.

.. data:: B160X160

160x160 resolution for the HM01B0 camera sensor.

.. data:: B320X320

320x320 resolution for the HM01B0 camera sensor.

.. data:: LCD

128x160 resolution for the camera sensor (for use with the lcd shield).

.. data:: QQVGA2

128x160 resolution for the camera sensor (for use with the lcd shield).

.. data:: WVGA

720x480 resolution for the MT9V034 camera sensor.

.. data:: WVGA2

752x480 resolution for the MT9V034 camera sensor.

.. data:: SVGA

800x600 resolution for the camera sensor.

.. data:: XGA

1024x768 resolution for the camera sensor.

.. data:: WXGA

1280x768 resolution for the MT9M114 camera sensor.

.. data:: SXGA

1280x1024 resolution for the camera sensor. Only works for the OV2640/OV5640 cameras.

.. data:: SXGAM

1280x960 resolution for the MT9M114 camera sensor.

.. data:: UXGA

1600x1200 resolution for the camera sensor. Only works for the OV2640/OV5640 cameras.

.. data:: HD

1280x720 resolution for the camera sensor.

.. data:: FHD

1920x1080 resolution for the camera sensor. Only works for the OV5640 camera.

.. data:: QHD

2560x1440 resolution for the camera sensor. Only works for the OV5640 camera.

.. data:: QXGA

2048x1536 resolution for the camera sensor. Only works for the OV5640 camera.

.. data:: WQXGA

2560x1600 resolution for the camera sensor. Only works for the OV5640 camera.

.. data:: WQXGA2

2592x1944 resolution for the camera sensor. Only works for the OV5640 camera.

.. data:: IOCTL\_SET\_READOUT\_WINDOW

Lets you set the readout window for the OV5640.

.. data:: IOCTL\_GET\_READOUT\_WINDOW

Lets you get the readout window for the OV5640.

.. data:: IOCTL\_SET\_TRIGGERED\_MODE

Lets you set the triggered mode for the MT9V034.

.. data:: IOCTL\_GET\_TRIGGERED\_MODE

Lets you get the triggered mode for the MT9V034.

.. data:: IOCTL\_SET\_FOV\_WIDE

Enable `sensor.set\_framesize()` to optimize for the field-of-view over FPS.

.. data:: IOCTL\_GET\_FOV\_WIDE

Return if `sensor.set\_framesize()` is optimizing for field-of-view over FPS.

.. data:: IOCTL\_TRIGGER\_AUTO\_FOCUS

Used to trigger auto focus for the OV5640 FPC camera module.

.. data:: IOCTL\_PAUSE\_AUTO\_FOCUS

Used to pause auto focus (while running) for the OV5640 FPC camera module.

.. data:: IOCTL\_RESET\_AUTO\_FOCUS

Used to reset auto focus back to the default for the OV5640 FPC camera module.

.. data:: IOCTL\_WAIT\_ON\_AUTO\_FOCUS

Used to wait on auto focus to finish after being triggered for the OV5640 FPC camera module.

.. data:: IOCTL\_SET\_NIGHT\_MODE

Used to turn night mode on or off on a sensor. Nightmode reduces the frame rate to increase exposure dynamically.

.. data:: IOCTL\_GET\_NIGHT\_MODE

Gets the current value of if night mode is enabled or disabled for your sensor.

.. data:: IOCTL\_LEPTON\_GET\_WIDTH

Lets you get the FLIR Lepton image resolution width in pixels.

.. data:: IOCTL\_LEPTON\_GET\_HEIGHT

Lets you get the FLIR Lepton image resolution height in pixels.

.. data:: IOCTL\_LEPTON\_GET\_RADIOMETRY

Lets you get the FLIR Lepton type (radiometric or not).

.. data:: IOCTL\_LEPTON\_GET\_REFRESH

Lets you get the FLIR Lepton refresh rate in hertz.

.. data:: IOCTL\_LEPTON\_GET\_RESOLUTION

Lets you get the FLIR Lepton ADC resolution in bits.

.. data:: IOCTL\_LEPTON\_RUN\_COMMAND

Executes a 16-bit command given the FLIR Lepton SDK.

.. data:: IOCTL\_LEPTON\_SET\_ATTRIBUTE

Sets a FLIR Lepton Attribute given the FLIR Lepton SDK.

.. data:: IOCTL\_LEPTON\_GET\_ATTRIBUTE

Gets a FLIR Lepton Attribute given the FLIR Lepton SDK.

.. data:: IOCTL\_LEPTON\_GET\_FPA\_TEMPERATURE

Gets the FLIR Lepton FPA temp in celsius.

.. data:: IOCTL\_LEPTON\_GET\_AUX\_TEMPERATURE

Gets the FLIR Lepton AUX temp in celsius.

.. data:: IOCTL\_LEPTON\_SET\_MEASUREMENT\_MODE

Lets you set the FLIR Lepton driver into a mode where you can get a valid temperature value per pixel. See `sensor.ioctl()` for more information.

.. data:: IOCTL\_LEPTON\_GET\_MEASUREMENT\_MODE

Lets you get if measurement mode is enabled or not for the FLIR Lepton sensor. See `sensor.ioctl()` for more information.

.. data:: IOCTL\_LEPTON\_SET\_MEASUREMENT\_RANGE

Lets you set the temperature range you want to map pixels in the image to when in measurement mode. See `sensor.ioctl()` for more information.

.. data:: IOCTL\_LEPTON\_GET\_MEASUREMENT\_RANGE

Lets you get the temperature range used for measurement mode. See `sensor.ioctl()` for more information.

.. data:: IOCTL\_HIMAX\_MD\_ENABLE

Lets you control the motion detection interrupt on the HM01B0. See `sensor.ioctl()` for more information.

.. data:: IOCTL\_HIMAX\_MD\_CLEAR

Lets you control the motion detection interrupt on the HM01B0. See `sensor.ioctl()` for more information.

.. data:: IOCTL\_HIMAX\_MD\_WINDOW

Lets you control the motion detection interrupt on the HM01B0. See `sensor.ioctl()` for more information.

.. data:: IOCTL\_HIMAX\_MD\_THRESHOLD

Lets you control the motion detection interrupt on the HM01B0. See `sensor.ioctl()` for more information.

.. data:: IOCTL\_HIMAX\_OSC\_ENABLE

Lets you control the internal oscillator on the HM01B0. See `sensor.ioctl()` for more information.

.. data:: SINGLE\_BUFFER

Pass to `sensor.set\_framebuffers()` to set single buffer mode (1 buffer).

.. data:: DOUBLE\_BUFFER

Pass to `sensor.set\_framebuffers()` to set double buffer mode (2 buffers).

.. data:: TRIPLE\_BUFFER

Pass to `sensor.set\_framebuffers()` to set triple buffer mode (3 buffers).

.. data:: VIDEO\_FIFO

Pass to `sensor.set\_framebuffers()` to set video FIFO mode (4 buffers).

#os库

:mod:`sensor` --- camera sensor

===============================

.. module:: sensor

:synopsis: camera sensor

The ``sensor`` module is used for taking pictures.

Example usage::

import sensor

# Setup camera.

sensor.reset()

sensor.set\_pixformat(sensor.RGB565)

sensor.set\_framesize(sensor.QVGA)

sensor.skip\_frames()

# Take pictures.

while(True):

sensor.snapshot()

Functions

---------

.. function:: reset()

Initializes the camera sensor.

.. function:: sleep(enable)

Puts the camera to sleep if enable is True. Otherwise, wakes it back up.

.. function:: shutdown(enable)

Puts the camera into a lower power mode than sleep (but the camera must be reset on being woken up).

.. function:: flush()

Copies whatever was in the frame buffer to the IDE. You should call this

method to display the last image your OpenMV Cam takes if it's not running

a script with an infinite loop. Note that you'll need to add a delay time

of about a second after your script finishes for the IDE to grab the image

from your camera. Otherwise, this method will have no effect.

.. function:: snapshot()

Takes a picture using the camera and returns an ``image`` object.

The OpenMV Cam has two memory areas for images. The classical stack/heap

area used for normal MicroPython processing can store small images within

it's heap. However, the MicroPython heap is only about ~100 KB which is not

enough to store larger images. So, your OpenMV Cam has a secondary frame

buffer memory area that stores images taken by `sensor.snapshot()`. Images

are stored on the bottom of this memory area. Any memory that's left

over is then available for use by the frame buffer stack which your OpenMV

Cam's firmware uses to hold large temporary data structures for image

processing algorithms.

If you need room to hold multiple frames you may "steal" frame buffer space

by calling `sensor.alloc\_extra\_fb()`.

If `sensor.set\_auto\_rotation()` is enabled this method will return a new

already rotated `image` object.

.. note::

`sensor.snapshot()` may apply cropping parameters to fit the snapshot in the available

RAM the pixformat, framesize, windowing, and framebuffers. The cropping parameters will be applied

to maintain the aspect ratio and will stay until `sensor.set\_framesize()` or `sensor.set\_windowing()` are called.

.. function:: skip\_frames([n, time])

Takes ``n`` number of snapshots to let the camera image stabilize after

changing camera settings. ``n`` is passed as normal argument, e.g.

``skip\_frames(10)`` to skip 10 frames. You should call this function after

changing camera settings.

Alternatively, you can pass the keyword argument ``time`` to skip frames

for some number of milliseconds, e.g. ``skip\_frames(time = 2000)`` to skip

frames for 2000 milliseconds.

If neither ``n`` nor ``time`` is specified this method skips frames for

300 milliseconds.

If both are specified this method skips ``n`` number of frames but will

timeout after ``time`` milliseconds.

.. note::

`sensor.snapshot()` may apply cropping parameters to fit the snapshot in the available

RAM given the pixformat, framesize, windowing, and framebuffers. The cropping parameters will be applied

to maintain the aspect ratio and will stay until `sensor.set\_framesize()` or `sensor.set\_windowing()` are called.

.. function:: width()

Returns the sensor resolution width.

.. function:: height()

Returns the sensor resolution height.

.. function:: get\_fb()

(Get Frame Buffer) Returns the image object returned by a previous call of

`sensor.snapshot()`. If `sensor.snapshot()` had not been called before

then ``None`` is returned.

.. function:: get\_id()

Returns the camera module ID.

\* `sensor.OV9650`: First gen OpenMV Cam sensor - never released.

\* `sensor.OV2640`: Second gen OpenMV Cam sensor - never released.

\* `sensor.OV5640`: High-res OpenMV Cam H7 sensor.

\* `sensor.OV7725`: Rolling shutter sensor module.

\* `sensor.OV7690`: OpenMV Cam Micro sensor module.

\* `sensor.MT9V034`: Global shutter sensor module.

\* `sensor.MT9M114`: New Rolling shutter sensor module.

\* `sensor.LEPTON`: Lepton1/2/3 sensor module.

\* `sensor.HM01B0`: Arduino Portenta H7 sensor module.

\* `sensor.GC2145`: Arduino Nicla Vision H7 sensor module.

\* `sensor.PAJ6100`: PixArt Imaging sensor Module.

.. function:: alloc\_extra\_fb(width, height, pixformat)

Allocates another frame buffer for image storage from the frame buffer stack

and returns an ``image`` object of ``width``, ``height``, and ``pixformat``.

You may call this function as many times as you like as long as there's

memory available to allocate any number of extra frame buffers.

If ``pixformat`` is a number >= 4 then this will allocate a JPEG image. You

can then do `Image.bytearray()` to get byte level read/write access to the JPEG image.

.. note::

Creating secondary images normally requires creating them on the heap which

has a limited amount of RAM... but, also gets fragmented making it hard to

grab a large contigous memory array to store an image in. With this method

you are able to allocate a very large memory array for an image instantly

by taking space away from our frame buffer stack memory which we use for

computer vision algorithms. That said, this also means you'll run out of

memory more easily if you try to execute more memory intensive machine

vision algorithms like `Image.find\_apriltags()`.

This is simplified version of a function which appears in the

``traceback`` module in CPython. Unlike ``traceback.print\_exception()``,

this function takes just exception value instead of exception type,

exception value, and traceback object; \*file\* argument should be

positional; further arguments are not supported. CPython-compatible

``traceback`` module can be found in `micropython-lib`.

.. function:: dealloc\_extra\_fb()

Deallocates the last previously allocated extra frame buffer. Extra frame

buffers are stored in a stack like structure.

.. note::

Your OpenMV Cam has two memory areas. First, you have your classical

.data/.bss/heap/stack memory area. The .data/.bss/heap regions are

fixed by firmware. The stack then grows down until it hits the heap.

Next, frame buffers are stored in a secondary memory region. Memory is

liad out with the main frame buffer on the bottom and the frame buffer

stack on the top. When `sensor.snapshot()` is called it fills the frame bufer

from the bottom. The frame buffer stack is then able to use whatever is

left over. This memory allocation method is extremely efficent for computer

vision on microcontrollers.

.. function:: set\_pixformat(pixformat)

Sets the pixel format for the camera module.

\* `sensor.GRAYSCALE`: 8-bits per pixel.

\* `sensor.RGB565`: 16-bits per pixel.

\* `sensor.BAYER`: 8-bits per pixel bayer pattern.

\* `sensor.YUV422`: 16-bits per pixel (8-bits Y1, 8-bits U, 8-bits Y2, 8-bits V, etc.)

\* `sensor.JPEG`: Compressed JPEG data. Only for the OV2640/OV5640.

If you are trying to take JPEG images with the OV2640 or OV5640 camera modules at high

resolutions you should set the pixformat to `sensor.JPEG`. You can control the image

quality then with `sensor.set\_quality()`.

.. function:: get\_pixformat()

Returns the pixformat for the camera module.

.. function:: set\_framesize(framesize)

Sets the frame size for the camera module.

\* `sensor.QQCIF`: 88x72

\* `sensor.QCIF`: 176x144

\* `sensor.CIF`: 352x288

\* `sensor.QQSIF`: 88x60

\* `sensor.QSIF`: 176x120

\* `sensor.SIF`: 352x240

\* `sensor.QQQQVGA`: 40x30

\* `sensor.QQQVGA`: 80x60

\* `sensor.QQVGA`: 160x120

\* `sensor.QVGA`: 320x240

\* `sensor.VGA`: 640x480

\* `sensor.HQQQQVGA`: 30x20

\* `sensor.HQQQVGA`: 60x40

\* `sensor.HQQVGA`: 120x80

\* `sensor.HQVGA`: 240x160

\* `sensor.HVGA`: 480x320

\* `sensor.B64X32`: 64x32 (for use with `Image.find\_displacement()`)

\* `sensor.B64X64`: 64x64 (for use with `Image.find\_displacement()`)

\* `sensor.B128X64`: 128x64 (for use with `Image.find\_displacement()`)

\* `sensor.B128X128`: 128x128 (for use with `Image.find\_displacement()`)

\* `sensor.B160X160`: 160x160 (for the HM01B0)

\* `sensor.B320X320`: 320x320 (for the HM01B0)

\* `sensor.LCD`: 128x160 (for use with the lcd shield)

\* `sensor.QQVGA2`: 128x160 (for use with the lcd shield)

\* `sensor.WVGA`: 720x480 (for the MT9V034)

\* `sensor.WVGA2`:752x480 (for the MT9V034)

\* `sensor.SVGA`: 800x600 (only for the OV2640/OV5640 sensor)

\* `sensor.XGA`: 1024x768 (only for the OV2640/OV5640 sensor)

\* `sensor.WXGA`: 1280x768 (for the MT9M114)

\* `sensor.SXGA`: 1280x1024 (only for the OV2640/OV5640 sensor)

\* `sensor.SXGAM`: 1280x960 (for the MT9M114)

\* `sensor.UXGA`: 1600x1200 (only for the OV2640/OV5640 sensor)

\* `sensor.HD`: 1280x720 (only for the OV2640/OV5640 sensor)

\* `sensor.FHD`: 1920x1080 (only for the OV5640 sensor)

\* `sensor.QHD`: 2560x1440 (only for the OV5640 sensor)

\* `sensor.QXGA`: 2048x1536 (only for the OV5640 sensor)

\* `sensor.WQXGA`: 2560x1600 (only for the OV5640 sensor)

\* `sensor.WQXGA2`: 2592x1944 (only for the OV5640 sensor)

.. function:: get\_framesize()

Returns the frame size for the camera module.

.. function:: set\_framerate(rate)

Sets the frame rate in hz for the camera module.

.. note::

`set\_framerate` works by dropping frames received by the camera module to keep the frame rate

equal to (or below) the rate you specify. By default the camera will run at the maximum frame

rate. If implemented for the particular camera sensor then `set\_framerate` will also reduce

the camera sensor frame rate internally to save power and improve image quality by increasing

the sensor exposure. `set\_framerate` may conflict with `set\_auto\_exposure` on some cameras.

.. function:: get\_framerate()

Returns the frame rate in hz for the camera module.

.. function:: set\_windowing(roi)

Sets the resolution of the camera to a sub resolution inside of the current

resolution. For example, setting the resolution to `sensor.VGA` and then

the windowing to (120, 140, 200, 200) sets `sensor.snapshot()` to capture

the 200x200 center pixels of the VGA resolution outputted by the camera

sensor. You can use windowing to get custom resolutions. Also, when using

windowing on a larger resolution you effectively are digital zooming.

``roi`` is a rect tuple (x, y, w, h). However, you may just pass (w, h) and

the ``roi`` will be centered on the frame. You may also pass roi not in parens.

This function will automatically handle cropping the passed roi to the framesize.

.. function:: get\_windowing()

Returns the ``roi`` tuple (x, y, w, h) previously set with `sensor.set\_windowing()`.

.. function:: set\_gainceiling(gainceiling)

Set the camera image gainceiling. 2, 4, 8, 16, 32, 64, or 128.

.. function:: set\_contrast(constrast)

Set the camera image contrast. -3 to +3.

.. function:: set\_brightness(brightness)

Set the camera image brightness. -3 to +3.

.. function:: set\_saturation(saturation)

Set the camera image saturation. -3 to +3.

.. function:: set\_quality(quality)

Set the camera image JPEG compression quality. 0 - 100.

.. note::

Only for the OV2640/OV5640 cameras.

.. function:: set\_colorbar(enable)

Turns color bar mode on (True) or off (False). Defaults to off.

.. function:: set\_auto\_gain(enable, [gain\_db=-1, [gain\_db\_ceiling]])

``enable`` turns auto gain control on (True) or off (False).

The camera will startup with auto gain control on.

If ``enable`` is False you may set a fixed gain in decibels with ``gain\_db``.

If ``enable`` is True you may set the maximum gain ceiling in decibels with

``gain\_db\_ceiling`` for the automatic gain control algorithm.

.. note::

You need to turn off white balance too if you want to track colors.

.. function:: get\_gain\_db()

Returns the current camera gain value in decibels (float).

.. function:: set\_auto\_exposure(enable, [exposure\_us])

``enable`` turns auto exposure control on (True) or off (False).

The camera will startup with auto exposure control on.

If ``enable`` is False you may set a fixed exposure time in microseconds

with ``exposure\_us``.

.. note::

Camera auto exposure algorithms are pretty conservative about how much

they adjust the exposure value by and will generally avoid changing the

exposure value by much. Instead, they change the gain value alot of deal

with changing lighting.

.. function:: get\_exposure\_us()

Returns the current camera exposure value in microseconds (int).

.. function:: set\_auto\_whitebal(enable, [rgb\_gain\_db])

``enable`` turns auto white balance on (True) or off (False).

The camera will startup with auto white balance on.

If ``enable`` is False you may set a fixed gain in decibels for the red, green,

and blue channels respectively with ``rgb\_gain\_db``.

.. note::

You need to turn off gain control too if you want to track colors.

.. function:: get\_rgb\_gain\_db()

Returns a tuple with the current camera red, green, and blue gain values in

decibels ((float, float, float)).

.. function:: set\_auto\_blc([enable, [regs]])

Sets the auto black line calibration (blc) control on the camera.

``enable`` pass `True` or `False` to turn BLC on or off. You typically always want this on.

``regs`` if disabled then you can manually set the blc register values via the values you

got previously from `get\_blc\_regs()`.

.. function:: get\_blc\_regs()

Returns the sensor blc registers as an opaque tuple of integers. For use with `set\_auto\_blc`.

.. function:: set\_hmirror(enable)

Turns horizontal mirror mode on (True) or off (False). Defaults to off.

.. function:: get\_hmirror()

Returns if horizontal mirror mode is enabled.

.. function:: set\_vflip(enable)

Turns vertical flip mode on (True) or off (False). Defaults to off.

.. function:: get\_vflip()

Returns if vertical flip mode is enabled.

.. function:: set\_transpose(enable)

Turns transpose mode on (True) or off (False). Defaults to off.

\* vflip=False, hmirror=False, transpose=False -> 0 degree rotation

\* vflip=True, hmirror=False, transpose=True -> 90 degree rotation

\* vflip=True, hmirror=True, transpose=False -> 180 degree rotation

\* vflip=False, hmirror=True, transpose=True -> 270 degree rotation

.. function:: get\_transpose()

Returns if transpose mode is enabled.

.. function:: set\_auto\_rotation(enable)

Turns auto rotation mode on (True) or off (False). Defaults to off.

.. note::

This function only works when the OpenMV Cam has an `imu` installed and is enabled automatically.

.. function:: get\_auto\_rotation()

Returns if auto rotation mode is enabled.

.. note::

This function only works when the OpenMV Cam has an `imu` installed and is enabled automatically.

.. function:: set\_framebuffers(count)

Sets the number of frame buffers used to receive image data. By default your OpenMV Cam will

automatically try to allocate the maximum number of frame buffers it can possibly allocate

without using more than 1/2 of the available frame buffer RAM at the time of allocation to

ensure the best performance. Automatic reallocation of frame buffers occurs whenever you

call `sensor.set\_pixformat()`, `sensor.set\_framesize()`, and `sensor.set\_windowing()`.

`sensor.snapshot()` will automatically handle switching active frame buffers in the background.

From your code's perspective there is only ever 1 active frame buffer even though there might

be more than 1 frame buffer on the system and another frame buffer reciving data in the background.

If count is:

1 - Single Buffer Mode (you may also pass `sensor.SINGLE\_BUFFER`)

In single buffer mode your OpenMV Cam will allocate one frame buffer for receiving images.

When you call `sensor.snapshot()` that framebuffer will be used to receive the image and

the camera driver will continue to run. In the advent you call `sensor.snapshot()` again

before the first line of the next frame is received your code will execute at the frame rate

of the camera. Otherwise, the image will be dropped.

2 - Double Buffer Mode (you may also pass `sensor.DOUBLE\_BUFFER`)

In double buffer mode your OpenMV Cam will allocate two frame buffers for receiving images.

When you call `sensor.snapshot()` one framebuffer will be used to receive the image and

the camera driver will continue to run. When the next frame is received it will be stored

in the other frame bufer. In the advent you call `sensor.snapshot()` again

before the first line of the next frame after is received your code will execute at the frame rate

of the camera. Otherwise, the image will be dropped.

3 - Triple Buffer Mode (you may also pass `sensor.TRIPLE\_BUFFER`)

In triple buffer mode your OpenMV Cam will allocate three buffers for receiving images.

In this mode there is always a frame buffer to store the received image to in the background

resulting in the highest performance and lowest latency for reading the latest received frame.

No frames are ever dropped in this mode. The next frame read by `sensor.snapshot()` is the

last captured frame by the sensor driver (e.g. if you are reading slower than the camera

frame rate then the older frame in the possible frames available is skipped).

Regarding the reallocation above, triple buffering is tried first, then double buffering, and if

these both fail to fit in 1/2 of the available frame buffer RAM then single buffer mode is used.

You may pass a value of 4 or greater to put the sensor driver into video FIFO mode where received

images are stored in a frame buffer FIFO with ``count`` buffers. This is useful for video recording

to an SD card which may randomly block your code from writing data when the SD card is performing

house-keeping tasks like pre-erasing blocks to write data to.

.. note::

On frame drop (no buffers available to receive the next frame) all frame buffers are automatically

cleared except the active frame buffer. This is done to ensure `sensor.snapshot()` returns current

frames and not frames from long ago.

Negative values of \*wbits\* between ``-5`` and ``-15`` correspond to "raw"

output mode, positive values between ``5`` and ``15`` correspond to zlib

output mode, and positive values between ``21`` and ``31`` correspond to

gzip output mode.

See the :mod:`CPython documentation for zlib <python:zlib>` for more

information about the \*wbits\* parameter. Note that MicroPython allows

for smaller window sizes, which is useful when memory is constrained while

still achieving a reasonable level of compression. It also speeds up

the compressor. See more :ref:`MicroPython-specific details <deflate\_wbits>`

in the :mod:`deflate <deflate>` module documentation.

Fun fact, you can pass a value of 100 or so on OpenMV Cam's with SDRAM for a huge video fifo. If

you then call snapshot slower than the camera frame rate (by adding `machine.sleep()`) you'll get

slow-mo effects in OpenMV IDE. However, you will also see the above policy effect of resetting

the frame buffer on a frame drop to ensure that frames do not get too old. If you want to record

slow-mo video just record video normally to the SD card and then play the video back on a desktop

machine slower than it was recorded.

.. function:: get\_framebuffers()

Returns the current number of frame buffers allocated.

.. function:: disable\_delays([disable])

If ``disable`` is ``True`` then disable all settling time delays in the sensor module.

Whenever you reset the camera module, change modes, etc. the sensor driver delays to prevent

you can from calling `snapshot` to quickly afterwards and receiving corrupt frames from the

camera module. By disabling delays you can quickly update the camera module settings in bulk

via multiple function calls before delaying at the end and calling `snapshot`.

If this function is called with no arguments it returns if delays are disabled.

.. function:: disable\_full\_flush([disable])

If ``disable`` is ``True`` then automatic framebuffer flushing mentioned in `set\_framebuffers`

is disabled. This removes any time limit on frames in the frame buffer fifo. For example, if

you set the number of frame buffers to 30 and set the frame rate to 30 you can now precisely

record 1 second of video from the camera without risk of frame loss.

If this function is called with no arguments it returns if automatic flushing is disabled. By

default automatic flushing on frame drop is enabled to clear out stale frames.

.. note::

`snapshot` starts the frame capture process which will continue to capture frames until

there is no space to hold a frame at which point the frame capture process stops. The

process always stops when there is no space to hold the next frame.

.. function:: set\_lens\_correction(enable, radi, coef)

``enable`` True to enable and False to disable (bool).

``radi`` integer radius of pixels to correct (int).

``coef`` power of correction (int).

.. function:: set\_vsync\_callback(cb)

Registers callback ``cb`` to be executed (in interrupt context) whenever the camera module

generates a new frame (but, before the frame is received).

``cb`` takes one argument and is passed the current state of the vsync pin after changing.

.. function:: set\_frame\_callback(cb)

Registers callback ``cb`` to be executed (in interrupt context) whenever the camera module

generates a new frame and the frame is ready to be read via `sensor.snapshot()`.

``cb`` takes no arguments.

Use this to get an interrupt to schedule reading a frame later with `micropython.schedule()`.

.. function:: get\_frame\_available()

Returns True if a frame is available to read by calling `sensor.snapshot()`.

.. function:: ioctl(...)

Executes a sensor specific method:

\* `sensor.IOCTL\_SET\_READOUT\_WINDOW` - Pass this enum followed by a rect tuple (x, y, w, h) or a size tuple (w, h).

\* This IOCTL allows you to control the readout window of the camera sensor which dramatically improves the frame rate at the cost of field-of-view.

\* If you pass a rect tuple (x, y, w, h) the readout window will be positoned on that rect tuple. The rect tuple's x/y position will be adjusted so the size w/h fits. Additionally, the size w/h will be adjusted to not be smaller than the ``framesize``.

\* If you pass a size tuple (w, h) the readout window will be centered given the w/h. Additionally, the size w/h will be adjusted to not be smaller than the ``framesize``.

\* This IOCTL is extremely helpful for increasing the frame rate on higher resolution cameras like the OV2640/OV5640.

\* `sensor.IOCTL\_GET\_READOUT\_WINDOW` - Pass this enum for `sensor.ioctl` to return the current readout window rect tuple (x, y, w, h). By default this is (0, 0, maximum\_camera\_sensor\_pixel\_width, maximum\_camera\_sensor\_pixel\_height).

\* `sensor.IOCTL\_SET\_TRIGGERED\_MODE` - Pass this enum followed by True or False set triggered mode for the MT9V034 sensor.

\* `sensor.IOCTL\_GET\_TRIGGERED\_MODE` - Pass this enum for `sensor.ioctl` to return the current triggered mode state.

\* `sensor.IOCTL\_SET\_FOV\_WIDE` - Pass this enum followed by True or False enable `sensor.set\_framesize()` to optimize for the field-of-view over FPS.

\* `sensor.IOCTL\_GET\_FOV\_WIDE` - Pass this enum for `sensor.ioctl` to return the current field-of-view over fps optimization state.

\* `sensor.IOCTL\_TRIGGER\_AUTO\_FOCUS` - Pass this enum for `sensor.ioctl` to trigger auto focus on the OV5640 FPC camera module.

\* `sensor.IOCTL\_PAUSE\_AUTO\_FOCUS` - Pass this enum for `sensor.ioctl` to pause auto focus (after triggering) on the OV5640 FPC camera module.

\* `sensor.IOCTL\_RESET\_AUTO\_FOCUS` - Pass this enum for `sensor.ioctl` to reset auto focus (after triggering) on the OV5640 FPC camera module.

\* `sensor.IOCTL\_WAIT\_ON\_AUTO\_FOCUS` - Pass this enum for `sensor.ioctl` to wait for auto focus (after triggering) to finish on the OV5640 FPC camera module. You may pass a second argument of the timeout in milliseconds. The default is 5000 ms.

\* `sensor.IOCTL\_SET\_NIGHT\_MODE` - Pass this enum followed by True or False set nightmode the OV7725 and OV5640 sensors.

\* `sensor.IOCTL\_GET\_NIGHT\_MODE` - Pass this enum for `sensor.ioctl` to return the current night mode state.

\* `sensor.IOCTL\_LEPTON\_GET\_WIDTH` - Pass this enum to get the FLIR Lepton image width in pixels.

\* `sensor.IOCTL\_LEPTON\_GET\_HEIGHT` - Pass this enum to get the FLIR Lepton image height in pixels.

\* `sensor.IOCTL\_LEPTON\_GET\_RADIOMETRY` - Pass this enum to get the FLIR Lepton type (radiometric or not).

\* `sensor.IOCTL\_LEPTON\_GET\_REFRESH` - Pass this enum to get the FLIR Lepton refresh rate in hertz.

\* `sensor.IOCTL\_LEPTON\_GET\_RESOLUTION` - Pass this enum to get the FLIR Lepton ADC resolution in bits.

\* `sensor.IOCTL\_LEPTON\_GET\_WIDTH` - Pass this enum to get the FLIR Lepton image width in pixels.

\* `sensor.IOCTL\_LEPTON\_GET\_HEIGHT` - Pass this enum to get the FLIR Lepton image height in pixels.

\* `sensor.IOCTL\_LEPTON\_RUN\_COMMAND` - Pass this enum to execute a FLIR Lepton SDK command. You need to pass an additional 16-bit value after the enum as the command to execute.

\* `sensor.IOCTL\_LEPTON\_SET\_ATTRIBUTE` - Pass this enum to set a FLIR Lepton SDK attribute.

\* The first argument is the 16-bit attribute ID to set (set the FLIR Lepton SDK).

\* The second argument is a MicroPython byte array of bytes to write (should be a multiple of 16-bits). Create the byte array using ``struct`` following the FLIR Lepton SDK.

\* `sensor.IOCTL\_LEPTON\_GET\_ATTRIBUTE` - Pass this enum to get a FLIR Lepton SDK attribute.

\* The first argument is the 16-bit attribute ID to set (set the FLIR Lepton SDK).

\* Returns a MicroPython byte array of the attribute. Use ``struct`` to deserialize the byte array following the FLIR Lepton SDK.

\* `sensor.IOCTL\_LEPTON\_GET\_FPA\_TEMPERATURE` - Pass this enum to get the FLIR Lepton FPA Temp in celsius.

\* `sensor.IOCTL\_LEPTON\_GET\_AUX\_TEMPERATURE` - Pass this enum to get the FLIR Lepton AUX Temp in celsius.

\* `sensor.IOCTL\_LEPTON\_SET\_MEASUREMENT\_MODE` - Pass this followed by True or False to turn off automatic gain control on the FLIR Lepton and force it to output an image where each pixel value represents an exact temperature value in celsius. A second True enables high temperature mode enabling measurements up to 500C on the Lepton 3.5, False is the default low temperature mode.

\* `sensor.IOCTL\_LEPTON\_GET\_MEASUREMENT\_MODE` - Pass this to get a tuple for (measurement-mode-enabled, high-temp-enabled).

\* `sensor.IOCTL\_LEPTON\_SET\_MEASUREMENT\_RANGE` - Pass this when measurement mode is enabled to set the temperature range in celsius for the mapping operation. The temperature image returned by the FLIR Lepton will then be clamped between these min and max values and then scaled to values between 0 to 255. To map a pixel value back to a temperature (on a grayscale image) do: ((pixel \* (max\_temp\_in\_celsius - min\_temp\_in\_celsius)) / 255.0) + min\_temp\_in\_celsius.

\* The first arugment should be the min temperature in celsius.

\* The second argument should be the max temperature in celsius. If the arguments are reversed the library will automatically swap them for you.

\* `sensor.IOCTL\_LEPTON\_GET\_MEASUREMENT\_RANGE` - Pass this to return the sorted (min, max) 2 value temperature range tuple. The default is -10C to 40C if not set yet.

\* `sensor.IOCTL\_HIMAX\_MD\_ENABLE` - Pass this enum followed by ``True``/``False`` to enable/disable motion detection on the HM01B0. You should also enable the I/O pin (PC15 on the Arduino Portenta) attached the HM01B0 motion detection line to receive an interrupt.

\* `sensor.IOCTL\_HIMAX\_MD\_CLEAR` - Pass this enum to clear the motion detection interrupt on the HM01B0.

\* `sensor.IOCTL\_HIMAX\_MD\_WINDOW` - Pass this enum followed by (x1, y1, x2, y2) to set the motion detection window on the HM01B0.

\* `sensor.IOCTL\_HIMAX\_MD\_THRESHOLD` - Pass this enum followed by a threshold value (0-255) to set the motion detection threshold on the HM01B0.

\* `sensor.IOCTL\_HIMAX\_OSC\_ENABLE` - Pass this enum followed by ``True``/``False`` to enable/disable the oscillator HM01B0 to save power.

.. function:: set\_color\_palette(palette)

Sets the color palette to use for FLIR Lepton grayscale to RGB565 conversion.

.. function:: get\_color\_palette()

Returns the current color palette setting. Defaults to `image.PALETTE\_RAINBOW`.

.. function:: \_\_write\_reg(address, value)

Write ``value`` (int) to camera register at ``address`` (int).

.. note:: See the camera data sheet for register info.

.. function:: \_\_read\_reg(address)

Read camera register at ``address`` (int).

.. note:: See the camera data sheet for register info.

Constants

-------------------------------------------------------------------------------------------------------------------------------------------

.. data:: BINARY

BINARY (bitmap) pixel format. Each pixel is 1-bit.

This format is usful for mask storage. Can be used with `image.Image()` and

`sensor.alloc\_extra\_fb()`.

.. data:: GRAYSCALE

GRAYSCALE pixel format (Y from YUV422). Each pixel is 8-bits, 1-byte.

All of our computer vision algorithms run faster on grayscale images than

RGB565 images.

.. data:: RGB565

RGB565 pixel format. Each pixel is 16-bits, 2-bytes. 5-bits are used for red,

6-bits are used for green, and 5-bits are used for blue.

All of our computer vision algorithms run slower on RGB565 images than

grayscale images.

.. data:: BAYER

RAW BAYER image pixel format. If you try to make the frame size too big

to fit in the frame buffer your OpenMV Cam will set the pixel format

to BAYER so that you can capture images but only some image processing methods

will be operational.

.. data:: YUV422

A pixel format that is very easy to jpeg compress. Each pixel is stored as a grayscale

8-bit Y value followed by alternating 8-bit U/V color values that are shared between two

Y values (8-bits Y1, 8-bits U, 8-bits Y2, 8-bits V, etc.). Only some image processing

methods work with YUV422.

.. data:: JPEG

JPEG mode. The camera module outputs compressed jpeg images.

Use `sensor.set\_quality()` to control the jpeg quality.

-------------------------------------------------------------------------------------------------------------------------------------------

Only works for the OV2640/OV5640 cameras.

#以下为ov2640镜头的调用函数

.. data:: OV2640

`sensor.get\_id()` returns this for the OV2640 camera.

#以下为ov5640镜头的调用函数

.. data:: OV5640

`sensor.get\_id()` returns this for the OV5640 camera.

#以下为ov7690镜头的调用函数

.. data:: OV7690

`sensor.get\_id()` returns this for the OV7690 camera.

#以下为ov7725镜头的调用函数

.. data:: OV7725

`sensor.get\_id()` returns this for the OV7725 camera.

#以下为ov9650镜头的调用函数

.. data:: OV9650

`sensor.get\_id()` returns this for the OV9650 camera.

#以下为MT9V022镜头的调用函数

.. data:: MT9V022

`sensor.get\_id()` returns this for the MT9V022 camera.

#以下为MT9V024镜头的调用函数

.. data:: MT9V024

`sensor.get\_id()` returns this for the MT9V024 camera.

#以下为MT9V032镜头的调用函数

.. data:: MT9V032

`sensor.get\_id()` returns this for the MT9V032 camera.

#以下为MT9V034镜头的调用函数

.. data:: MT9V034

`sensor.get\_id()` returns this for the MT9V034 camera.

#以下为MT9M114镜头的调用函数

.. data:: MT9M114

`sensor.get\_id()` returns this for the MT9M114 camera.

#以下为LEPTON镜头的调用函数

.. data:: LEPTON

`sensor.get\_id()` returns this for the LEPTON1/2/3 cameras.

#以下为HM01B0镜头的调用函数

.. data:: HM01B0

`sensor.get\_id()` returns this for the HM01B0 camera.

#以下为HM0360的调用函数

.. data:: HM0360

`sensor.get\_id()` returns this for the HM01B0 camera.

#以下为GC2145的调用函数

.. data:: GC2145

`sensor.get\_id()` returns this for the GC2145 camera.

#以下为PAJ6100的调用函数

.. data:: PAJ6100

`sensor.get\_id()` returns this for the PAJ6100 camera.

#以下为FROGEYE2020的调用函数

.. data:: FROGEYE2020

`sensor.get\_id()` returns this for the FROGEYE2020 camera.

#以下为QQCIF的调用函数

.. data:: QQCIF

88x72 resolution for the camera sensor.

#以下为QCIF的调用函数

.. data:: QCIF

176x144 resolution for the camera sensor.

#以下为CIF的调用函数

.. data:: CIF

352x288 resolution for the camera sensor.

#以下为QQSIF的调用函数

.. data:: QQSIF

88x60 resolution for the camera sensor.

#以下为QSIF的调用函数

.. data:: QSIF

176x120 resolution for the camera sensor.

#以下为SIF的调用函数

.. data:: SIF

352x240 resolution for the camera sensor.

#以下为QQQQVGA的调用函数

.. data:: QQQQVGA

40x30 resolution for the camera sensor.

#以下为QQQVGA的调用函数

.. data:: QQQVGA

80x60 resolution for the camera sensor.

#以下为QQVGA的调用函数

.. data:: QQVGA

160x120 resolution for the camera sensor.

#以下为QVGA 的调用函数

.. data:: QVGA

320x240 resolution for the camera sensor.

#以下为VGA 的调用函数

.. data:: VGA

640x480 resolution for the camera sensor.

#以下为HQQQQVGA的调用函数

.. data:: HQQQQVGA

30x20 resolution for the camera sensor.

#以下为HQQQVGA的调用函数

.. data:: HQQQVGA

60x40 resolution for the camera sensor.

#以下为HQQVGA的调用函数

.. data:: HQQVGA

120x80 resolution for the camera sensor.

#以下为HQVGA的调用函数

.. data:: HQVGA

240x160 resolution for the camera sensor.

#以下为HVGA的调用函数

.. data:: HVGA

480x320 resolution for the camera sensor.

#以下为B64X32的调用函数

.. data:: B64X32

64x32 resolution for the camera sensor.

For use with `Image.find\_displacement()` and any other FFT based algorithm.

-------------------------------------------------------------------------------------------------------------------------------------------

.. data:: B64X64

64x64 resolution for the camera sensor.

For use with `Image.find\_displacement()` and any other FFT based algorithm.

.. data:: B128X64

128x64 resolution for the camera sensor.

For use with `Image.find\_displacement()` and any other FFT based algorithm.

.. data:: B128X128

128x128 resolution for the camera sensor.

For use with `Image.find\_displacement()` and any other FFT based algorithm.

.. data:: B160X160

160x160 resolution for the HM01B0 camera sensor.

.. data:: B320X320

320x320 resolution for the HM01B0 camera sensor.

.. data:: LCD

128x160 resolution for the camera sensor (for use with the lcd shield).

.. data:: QQVGA2

128x160 resolution for the camera sensor (for use with the lcd shield).

.. data:: WVGA

720x480 resolution for the MT9V034 camera sensor.

.. data:: WVGA2

752x480 resolution for the MT9V034 camera sensor.

.. data:: SVGA

800x600 resolution for the camera sensor.

.. data:: XGA

1024x768 resolution for the camera sensor.

.. data:: WXGA

1280x768 resolution for the MT9M114 camera sensor.

.. data:: SXGA

1280x1024 resolution for the camera sensor. Only works for the OV2640/OV5640 cameras.

.. data:: SXGAM

1280x960 resolution for the MT9M114 camera sensor.

.. data:: UXGA

1600x1200 resolution for the camera sensor. Only works for the OV2640/OV5640 cameras.

.. data:: HD

1280x720 resolution for the camera sensor.

.. data:: FHD

1920x1080 resolution for the camera sensor. Only works for the OV5640 camera.

.. data:: QHD

2560x1440 resolution for the camera sensor. Only works for the OV5640 camera.

.. data:: QXGA

2048x1536 resolution for the camera sensor. Only works for the OV5640 camera.

.. data:: WQXGA

2560x1600 resolution for the camera sensor. Only works for the OV5640 camera.

.. data:: WQXGA2

2592x1944 resolution for the camera sensor. Only works for the OV5640 camera.

-------------------------------------------------------------------------------------------------------------------------------------------

.. data:: IOCTL\_SET\_READOUT\_WINDOW

Lets you set the readout window for the OV5640.

.. data:: IOCTL\_GET\_READOUT\_WINDOW

Lets you get the readout window for the OV5640.

.. data:: IOCTL\_SET\_TRIGGERED\_MODE

Lets you set the triggered mode for the MT9V034.

.. data:: IOCTL\_GET\_TRIGGERED\_MODE

Lets you get the triggered mode for the MT9V034.

.. data:: IOCTL\_SET\_FOV\_WIDE

Enable `sensor.set\_framesize()` to optimize for the field-of-view over FPS.

.. data:: IOCTL\_GET\_FOV\_WIDE

Return if `sensor.set\_framesize()` is optimizing for field-of-view over FPS.

.. data:: IOCTL\_TRIGGER\_AUTO\_FOCUS

Used to trigger auto focus for the OV5640 FPC camera module.

.. data:: IOCTL\_PAUSE\_AUTO\_FOCUS

Used to pause auto focus (while running) for the OV5640 FPC camera module.

.. data:: IOCTL\_RESET\_AUTO\_FOCUS

Used to reset auto focus back to the default for the OV5640 FPC camera module.

.. data:: IOCTL\_WAIT\_ON\_AUTO\_FOCUS

Used to wait on auto focus to finish after being triggered for the OV5640 FPC camera module.

.. data:: IOCTL\_SET\_NIGHT\_MODE

Used to turn night mode on or off on a sensor. Nightmode reduces the frame rate to increase exposure dynamically.

------------------------------------------------------------------------------------------------------------------------------------------

.. data:: IOCTL\_GET\_NIGHT\_MODE

Gets the current value of if night mode is enabled or disabled for your sensor.

.. data:: IOCTL\_LEPTON\_GET\_WIDTH

Lets you get the FLIR Lepton image resolution width in pixels.

.. data:: IOCTL\_LEPTON\_GET\_HEIGHT

Lets you get the FLIR Lepton image resolution height in pixels.

.. data:: IOCTL\_LEPTON\_GET\_RADIOMETRY

Lets you get the FLIR Lepton type (radiometric or not).

.. data:: IOCTL\_LEPTON\_GET\_REFRESH

Lets you get the FLIR Lepton refresh rate in hertz.

.. data:: IOCTL\_LEPTON\_GET\_RESOLUTION

Lets you get the FLIR Lepton ADC resolution in bits.

.. data:: IOCTL\_LEPTON\_RUN\_COMMAND

Executes a 16-bit command given the FLIR Lepton SDK.

.. data:: IOCTL\_LEPTON\_SET\_ATTRIBUTE

Sets a FLIR Lepton Attribute given the FLIR Lepton SDK.

.. data:: IOCTL\_LEPTON\_GET\_ATTRIBUTE

Gets a FLIR Lepton Attribute given the FLIR Lepton SDK.

.. data:: IOCTL\_LEPTON\_GET\_FPA\_TEMPERATURE

Gets the FLIR Lepton FPA temp in celsius.

.. data:: IOCTL\_LEPTON\_GET\_AUX\_TEMPERATURE

Gets the FLIR Lepton AUX temp in celsius.

----------------------------------------------------------------------------------------------------------------------------------------

.. data:: IOCTL\_LEPTON\_SET\_MEASUREMENT\_MODE

Lets you set the FLIR Lepton driver into a mode where you can get a valid temperature value per pixel. See `sensor.ioctl()` for more information.

.. data:: IOCTL\_LEPTON\_GET\_MEASUREMENT\_MODE

Lets you get if measurement mode is enabled or not for the FLIR Lepton sensor. See `sensor.ioctl()` for more information.

.. data:: IOCTL\_LEPTON\_SET\_MEASUREMENT\_RANGE

Lets you set the temperature range you want to map pixels in the image to when in measurement mode. See `sensor.ioctl()` for more information.

-------------------------------------------------------------------------------------------------------------------------------------------

.. data:: IOCTL\_LEPTON\_GET\_MEASUREMENT\_RANGE

Lets you get the temperature range used for measurement mode. See `sensor.ioctl()` for more information.

.. data:: IOCTL\_HIMAX\_MD\_ENABLE

Lets you control the motion detection interrupt on the HM01B0. See `sensor.ioctl()` for more information.

.. data:: IOCTL\_HIMAX\_MD\_CLEAR

Lets you control the motion detection interrupt on the HM01B0. See `sensor.ioctl()` for more information.

.. data:: IOCTL\_HIMAX\_MD\_WINDOW

Lets you control the motion detection interrupt on the HM01B0. See `sensor.ioctl()` for more information.

.. data:: IOCTL\_HIMAX\_MD\_THRESHOLD

Lets you control the motion detection interrupt on the HM01B0. See `sensor.ioctl()` for more information.

.. data:: IOCTL\_HIMAX\_OSC\_ENABLE

Lets you control the internal oscillator on the HM01B0. See `sensor.ioctl()` for more information.

---------------------------------------------------------------------------------------------------------------------------------------

.. data:: SINGLE\_BUFFER

Pass to `sensor.set\_framebuffers()` to set single buffer mode (1 buffer).

.. data:: DOUBLE\_BUFFER

Pass to `sensor.set\_framebuffers()` to set double buffer mode (2 buffers).

.. data:: TRIPLE\_BUFFER

Pass to `sensor.set\_framebuffers()` to set triple buffer mode (3 buffers).

.. data:: VIDEO\_FIFO

Pass to `sensor.set\_framebuffers()` to set video FIFO mode (4 buffers).

#库函数部分到此结束

-------------------------------------------------------------------------------------------------------------------------------------------

#all end