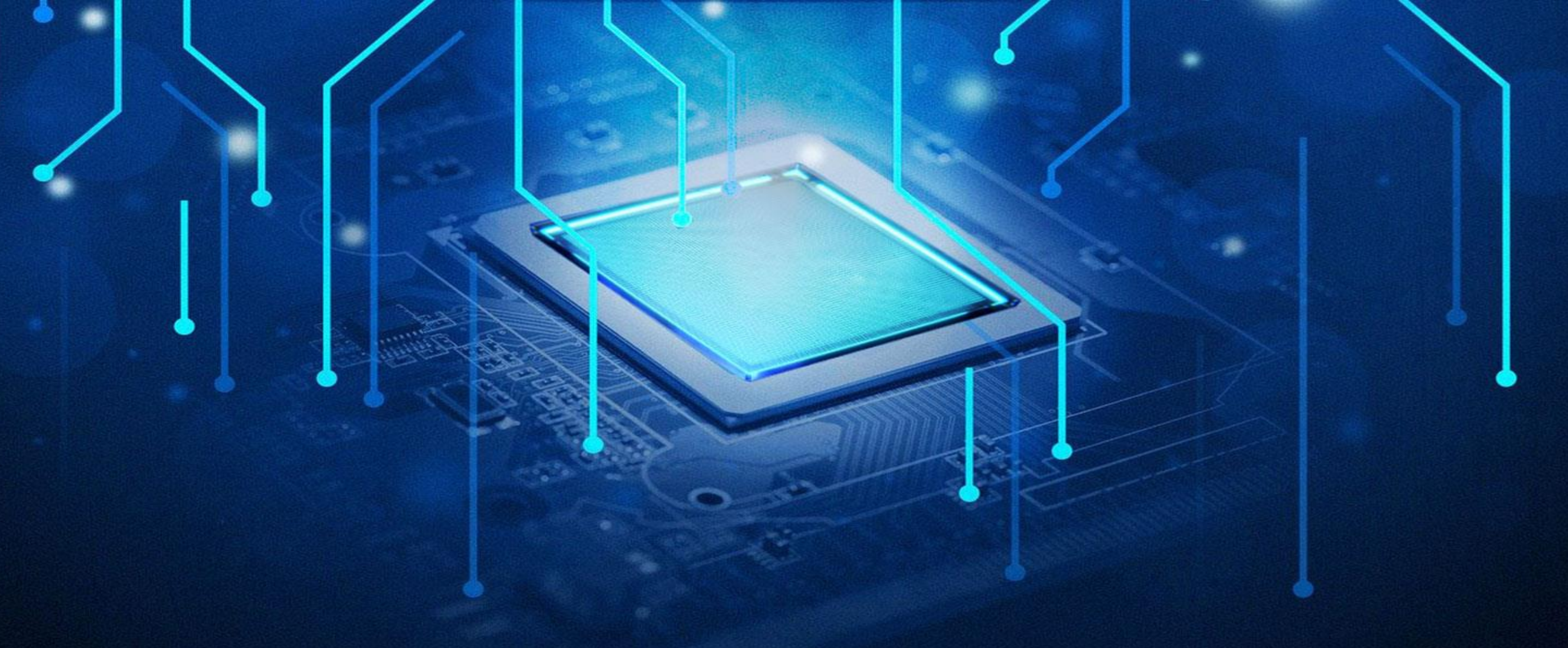


AI for EDGE

TinyML for edge computing

Outline

- TinyML
  - Motivation
  - Techniques
  - Benefits
- MicroPython
- Baremetal AI
- ESP project

How many transistors does it cost to implement a single multiplication in a network?

1. As a dedicated digital circuit:
   - m.in. 38 transistors/bit
2. As a reconfigurable digital circuit:
   - 32k transistors/weight
     (in 40 LUTs)

TinyML - motivation

- The average consumer CPU will draw between 65 to 85 watts of power, while the average GPU consumes anywhere between 200 to 500 watts.
- A typical microcontroller draws power in the order of milliwatts or microwatts, which is a thousand times less power consumption. This energy efficiency enables the TinyML devices to run on battery power while running ML applications on the edge.
- TinyML with its support for frameworks that include TensorFlow Lite, uTensor, and Arm's CMSIS-NN, brings together AI and small connected devices.

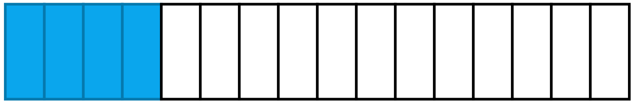
Definition: *This is a concept of implementing mainly deep neural networks directly on embedded devices with highly limited resources.*

This approach may include:

- adapting (reducing) the network architecture to reduced hardware resources
- reduction of power consumption during processing (e.g. for use in battery powered applications)

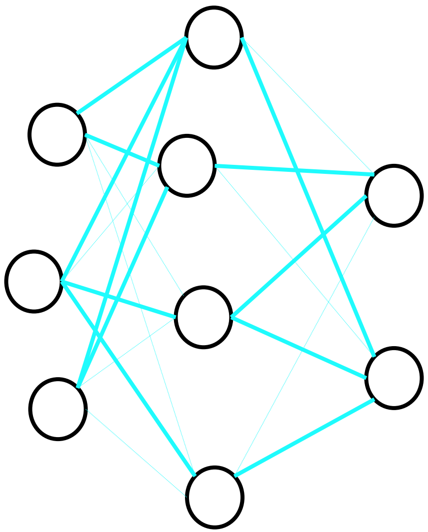
TinyML can be understood as a network algorithm compression method.

Quantization:



- The default representation of weights in the model is **32-bit floating point** numbers
- Quantization reduces the accuracy to **8-bit integers**

  - The model running on the processor after quantization runs faster
  - The technique is dedicated to devices with small memory
  - It brings special effects for complex models, i.e. those that have a lot of weights.

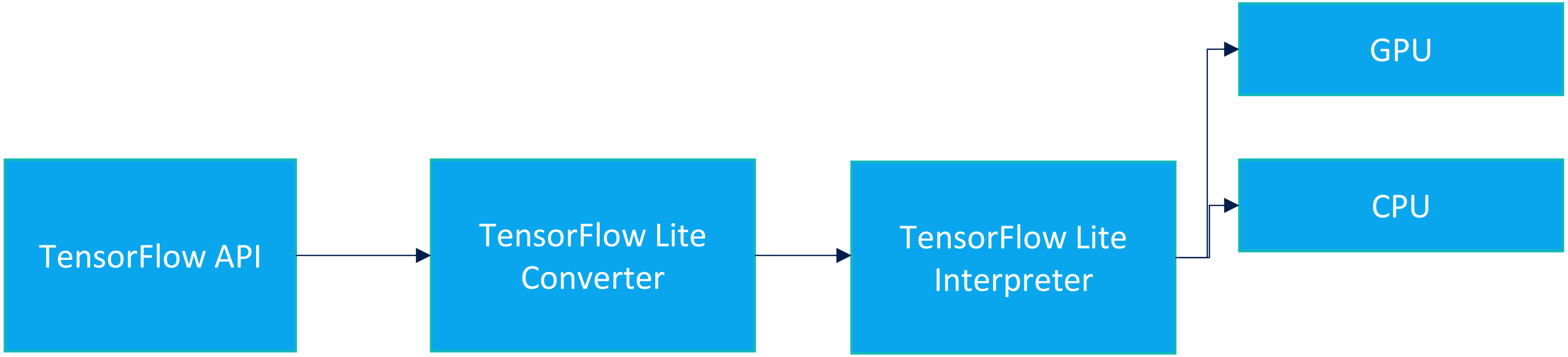


Pruning:

- It involves cutting parameters for reduction model size
- Results in deterioration of model parameters
- It usually requires iterative modification of the model
- It is difficult to define a universal, i.e. model-independent pruning method

TinyML - tools

- TensorFlow Lite Converter – to convert a model imported from the TensorFlow format
- TensorFlow Lite Interpreter – to load the finished model to the microcontroller's memory



TensorFlow Lite – conversion process

Input models:

- Saved model - classic TF model on disk
  - Keras H5 format – hierarchical data format HDF5
- Keras model – based on API Keras high-level interface
- models built of functions - based on API Keras low-level interface

The resulting TensorFlow Lite model:

.tflite – FlatBuffer format

Conversion options:

- Compliance options – permission to use operators
- Optimization options – defining the optimization used for the conversion
- Metadata options – adding metadata to the model

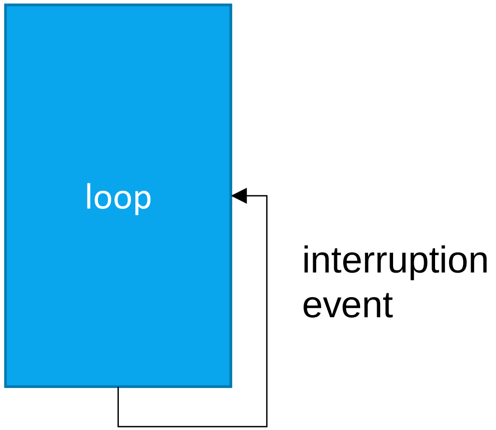
Benefits of TinyML

- **Energy efficiency**: Microcontrollers consume very little power, which delivers benefits in remote installations and mobile devices.
- **Low latency**: By processing data locally at the edge, data doesn't need to be transmitted to the cloud for inference. This greatly reduces device latency.
- **Privacy**: Data can be stored locally, not on cloud servers.
- **Reduced bandwidth**: With decreased dependency on the cloud for inference, bandwidth concerns are minimized.

- The future of TinyML using MCUs is promising for small edge devices and modest applications where an FPGA, GPU or CPU are not viable options.

Definition: An approach based on programming the application directly "on the hardware" (with direct access to the microcontroller registers), i.e. without using a programming interface, e.g. an operating system.

One of the more commonly used bare-metal implementations is the infinite-time super-loop that the microcontroller executes. The execution of the loop is stopped by an interruption event.



RTOS:

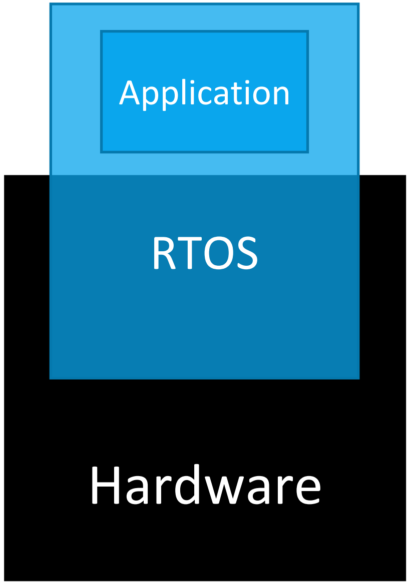
System kernel with scheduler

Device drivers

Multithreading

Prioritizing tasks

Most likely a quick starting point



BARE METAL:

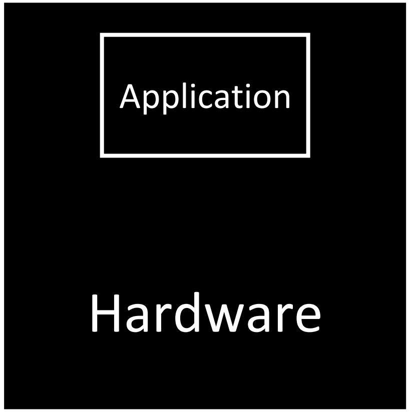
Customized solution

Solution planned in detal

Lower costs of software execution

Most likely a longer starting point

Harder to develop project in the future



MicroPython

- A lightweight version of the Python 3 programming language
- A subset of the Python standard library
- Optimized to work with microcontrollers
- Requirements: 256 KB for code and 16 KB of RAM
- The functionality includes:
  - Integers of arbitrary precision
  - Interactive prompt
  - Exception handling
  - Comprehension letter

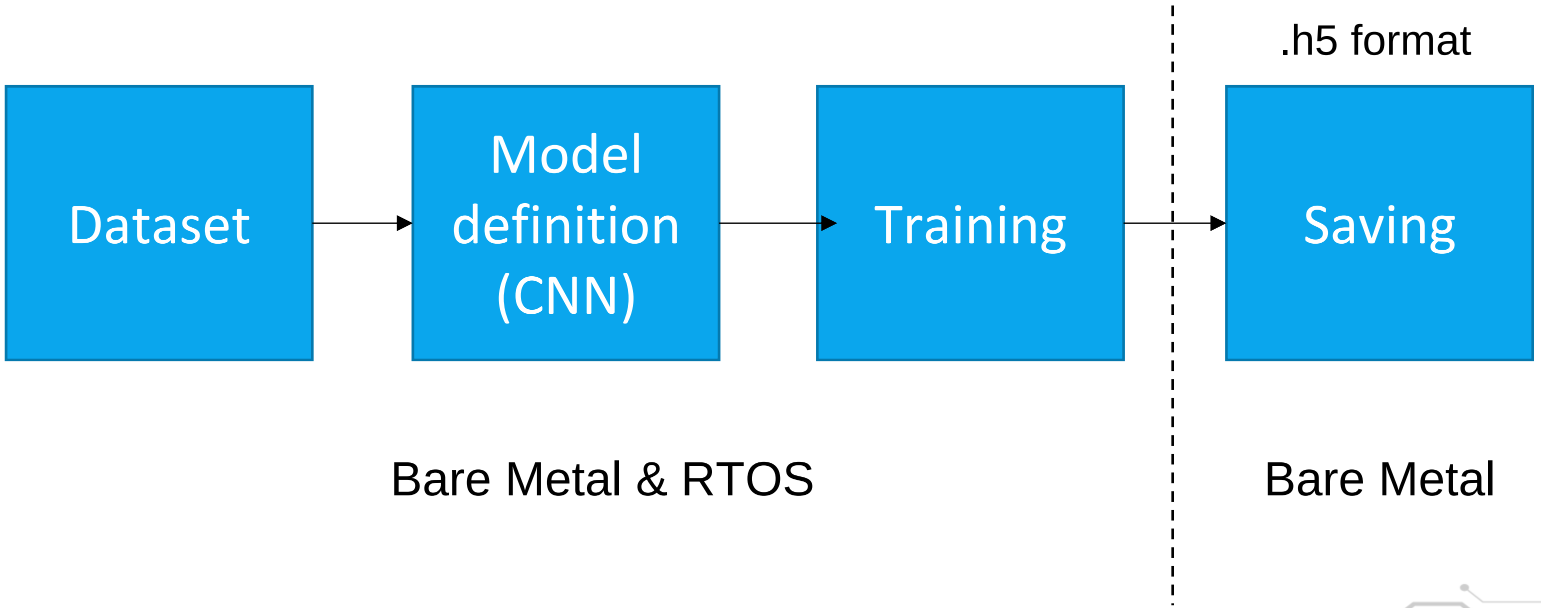
https://micropython.org/

Porty: cc3200, esp32, esp8266, mimxrt, nrf,
renesas-ra, rp2, samd, stm32

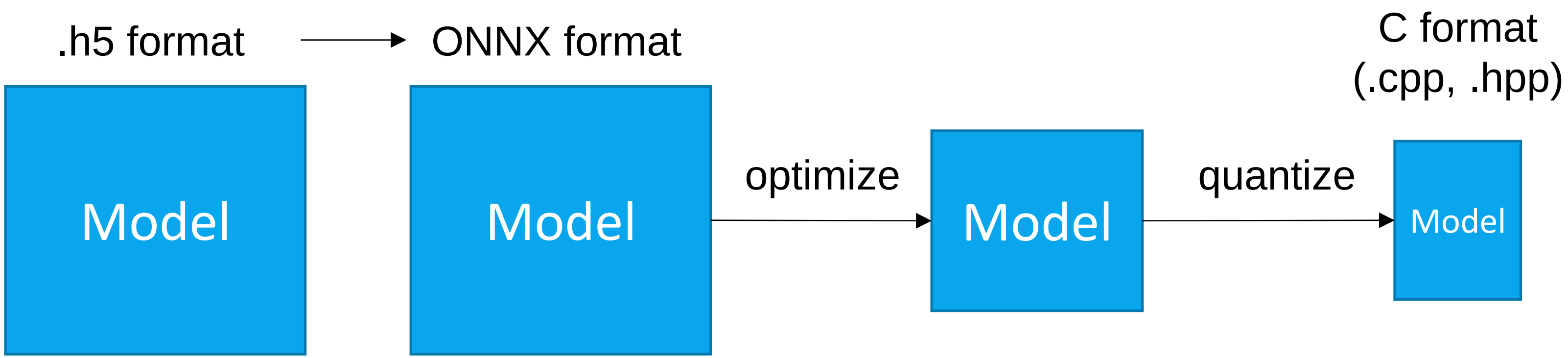
ESP32-CAM specification

- Clock frequency: up to 160 MHz
- **520KB RAM**
- **512 kB FLASH**
- WiFi 802.11 b/g/n module
- Security: TKIP, WEP, CRC, CCMP, WPA/WPA2, WPS
- UART/SPI/I2C/PWM/ADC/DAC interfaces
- OV2640 camera (2 MPx resolution)
- Possibility to connect the OV2640 or OV7670 camera
- microSD slot (up to 4GB)
- User LED
- RESET button

1. Prepare the model



2. ESP conversion



!python -m tf2onnx.convert.py --input %name% --inputs %input_name% --outputs %output_name%

calib = Calibrator('int16', 'per-tensor', 'minmax')

Bare Metal AI (based on ESP)

3. ESP project

- Components
- Model (.cpp, .hpp)
- Sdk config
- Librariers

ESP-DL
https://github.com/espressif/esp-dl
* **Model Zoo**

ESP-WHO
https://github.com/espressif/esp-who

Related topics

- TinyML is bringing deep learning models to microcontrollers
  https://thenextweb.com/news/tinyml-deep-learning-microcontrollers-syndication
- ESP32-CAM: TinyML Image Classification
  https://mjrobot.org/2022/02/10/esp32-cam-tinyml-image-classification-fruits-vs-veggies/