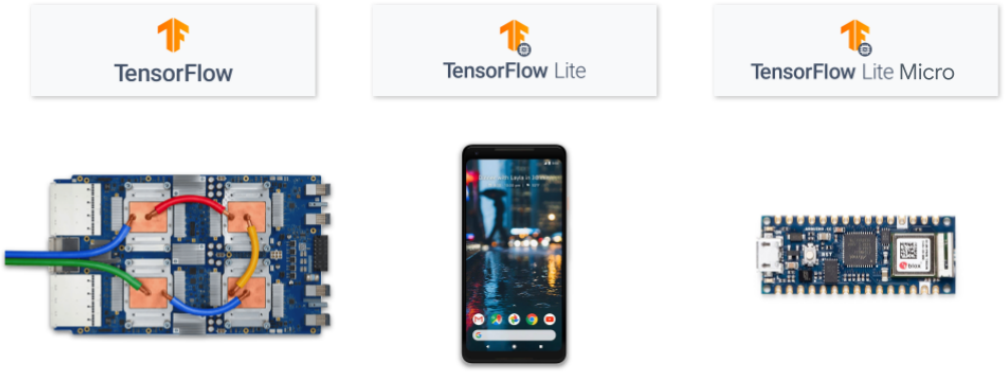
# TFLite Micro for TinyML




## Motivation & Background

TensorFlow, developed by Google, is a very popular machine learning framework for constructing and running machine learning models, and we have used it several times in this series of modules to construct our machine learning models. However, there are several downsides to TensorFlow that make it unsuitable for use in microcontroller units (MCUs).

Most notably, it is just too darn big! For starters, downloading TensorFlow on your laptop or desktop computer will take up around 1 GB of hard disk space - on microcontrollers, we are lucky if we have 1 MB of space! Next, the RAM required to run even the most basic of models is too high for any MCU, which typically have RAM of 500 kB or less. TensorFlow also works with floating point arithmetic by default (FP16/FP32), which is not supported on most MCUs due to the lack of a floating point unit (FPU). Most MCUs only have access to an arithmetic logic unit (ALU) for performing 8-bit integer operations.

There is a happy ending to this story though. The TensorFlow team at Google recognised that there is a market for running machine learning models on resource-constrained devices that would require a more lightweight machine learning framework. Enter TensorFlow Lite (TFLite). TFLite is a framework for deploying ML models on multiple devices ranging from mobile (iOS and Android), desktop, and edge devices. An even more lightweight version exists specifically for microcontrollers, called TFLite Micro (sometimes called TFMicro).

## TFLite Micro Overview

TFLite Micro is a barebones ML inference engine written in C++11 which acts as a stripped out version of the TensorFlow library, removing all unnecessary components such as stored models, debugging capabilities, and unnecessary operations. The team, headed by Pete Warden, managed to compress the core runtime into approximately 16 kB (on an ARM Cortex M3), which is sufficient to run on a large number of resource-constrained devices. This is incredibly impressive given some of the additional challenges associated with resource-constrained

devices. Note that TFLite Micro is only an inference engine, and thus does not natively support on-device training.

The heterogeneity of embedded systems makes interoperability across the landscape of embedded devices challenging. As discussed previously, most MCUs cannot handle floating point operations. However, some can. Similarly, some work with 16-bit integers as opposed to 8-bit. Their architectures often vary, as do their peripherals and the presence of other computational components. This makes it incredibly difficult to design a framework that is configurable with all types of heterogeneity environments where devices have no common standard. When we download TensorFlow on our laptop or desktop computer, we have to specify what system we are using. Is it 32-bit or 64-bit? Is it running MacOS, Windows, or some form of Linux? Which version is the operating system? For microcontrollers, we come across the same issues, but the number of options is far greater due to the lack of a common standard. Currently, the only constraint imposed by TFLite Micro is the need for a 32-bit platform. The framework has been tested extensively on the ARM Cortex-M series of microcontrollers, and has also been ported to ESP32 and can be downloaded as an Arduino library. Boards currently supported by the framework include the following and this is continuously growing:

- Arduino Nano 33 BLE Sense
- SparkFun Edge
- STM32F746 Discovery kit
- Adafruit EdgeBadge
- Adafruit TensorFlow Lite for Microcontrollers Kit
- Adafruit Circuit Playground Bluefruit
- Espressif ESP32-DevKitC
- Espressif ESP-EYE
- Wio Terminal: ATSAMD51
- Himax WE-I Plus EVB Endpoint AI Development Board
- Synopsys DesignWare ARC EM Software Development Platform
- Sony Spresense

There is currently a trade-off between offering interoperability and performance. Because no industry standard currently exist, making TFLite Micro increasingly portable would require more code, which will inevitably make the library larger. Currently, the library only offers support for a limited set of devices (listed above), a limited set of TensorFlow operations (addition, subtraction, matrix multiplication, ReLU, etc.), and requires manual memory management (since the library does not utilize dynamic memory allocation).

Other TinyML frameworks are gradually coming into existence, most notably Pytorch's GLOW which utilizes the GLOW ahead of time (AOT) compiler. TFLite Micro seems to be the dominant framework currently, but this could change in the future. For a deeper dive into TFLite Micro and its structure, capabilities, and limitations, we refer the reader to the origin TFLite Micro research paper available on arXiv.