TensorFlow Lite for Microcontrollers: Recent Developments

Advait Jain
Staff Software Engineer
Google

David Davis
Senior Embedded Software Engineer
BDTI

John Withers
Automation and Systems Engineer
BDTI

Outline

- 10,000-foot view of TensorFlow Lite Micro
- BDTI/Google Collaboration
  - Updated Arduino port of TFLM
  - New Kernel Operators
  - Improved CI via GitHub Actions

TensorFlow Family (10,000-Foot view)

- TensorFlow (platform & ecosystem)
  - End-to-end open source platform for machine learning
  - Comprehensive, flexible ecosystem of tools, libraries and community resources that lets researchers push the state-of-the-art in ML and developers easily build and deploy ML powered applications
- TensorFlow (library)
  - The core open source library to help you develop and train ML models
- TensorFlow Lite
  - Library for deploying models on mobile, microcontrollers and other edge devices
- TensorFlow Lite Micro (TFLM)
  - Library to run machine learning models on DSPs, microcontrollers, and other embedded targets with a small memory footprint and very low power usage

TensorFlow Lite Micro (10,000-Foot View)

- Library designed to run machine learning models on embedded targets without any OS support, no dynamic memory allocation and a reduced set of C++11 standard libraries
- Leverages the model optimization tools from the TensorFlow ecosystem and has additional embedded-specific offline and online optimizations to reduce the memory footprint from both the model and the framework
- Integrates with a number of community contributed highly-optimized hardware-specific kernel implementations
- All the TFLM modules are tested on a variety of targets and toolchains via software emulation for each pull request to the TFLM GitHub repository
- TFLM provides tools, CI, and examples for how to integrate it into various embedded development environments

BDTI/Google Collaboration: Updated TFLM Port to Arduino

TFLM Arduino Examples: New Repository

Google and BDTI have created a new repository with platform specific example code for the Arduino Nano 33 BLE Sense.

The code base is synchronized nightly from the TFLM repository using Github workflows.

All example applications are independently maintained within the Arduino examples repository.

Includes support for CMSIS_NN.

Code in this repository can be used with both the Arduino IDE and CLI. With single step Git cloning of the repository into the Arduino library folder, TFLM is ready for use.

TFLM Arduino Examples: New Repository (cont.)

Repository adheres to this guideline document:

https://github.com/tensorflow/tflite-micro/blob/main/tensorflow/lite/micro/docs/new_platform_support.md

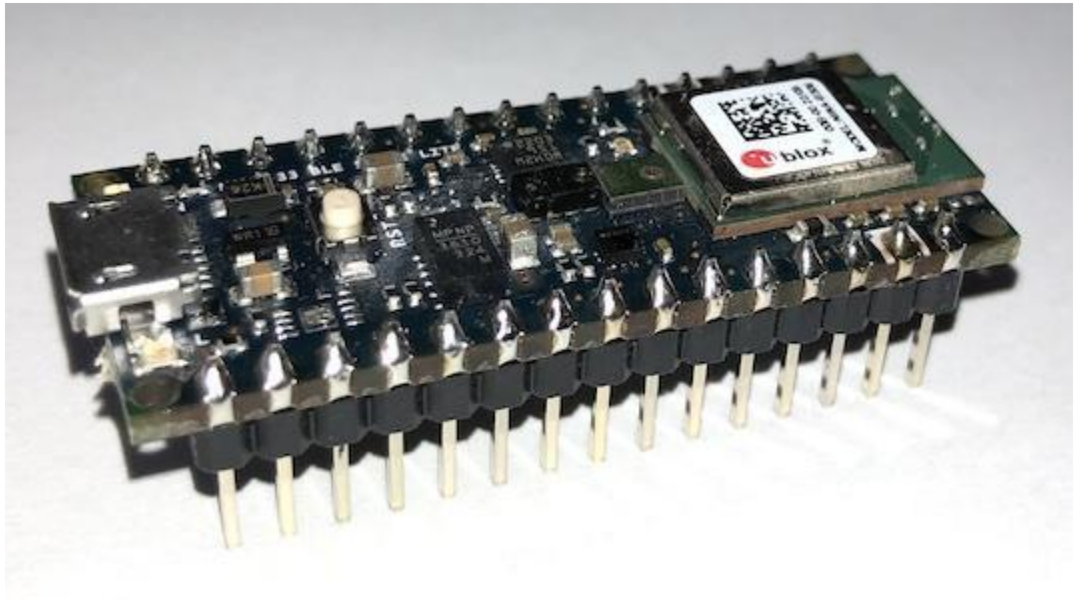
Nightly synchronization script and workflow:

https://github.com/tensorflow/tflite-micro-arduino-examples/blob/main/scripts/sync_from_tflite_micro.sh

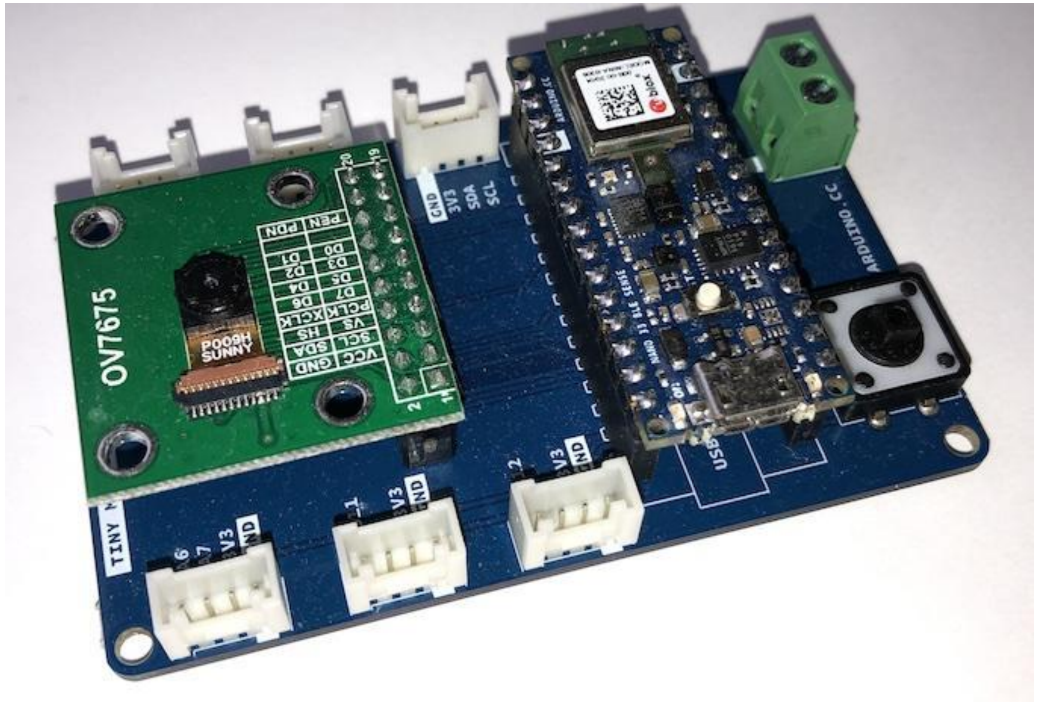
https://github.com/tensorflow/tflite-micro-arduino-examples/blob/main/.github/workflows/sync.yml

TFLM Arduino Examples: Tested Devices

Arduino Nano 33 BLE Sense



Tiny Machine Learning Kit
(with Nano 33 BLE Sense)



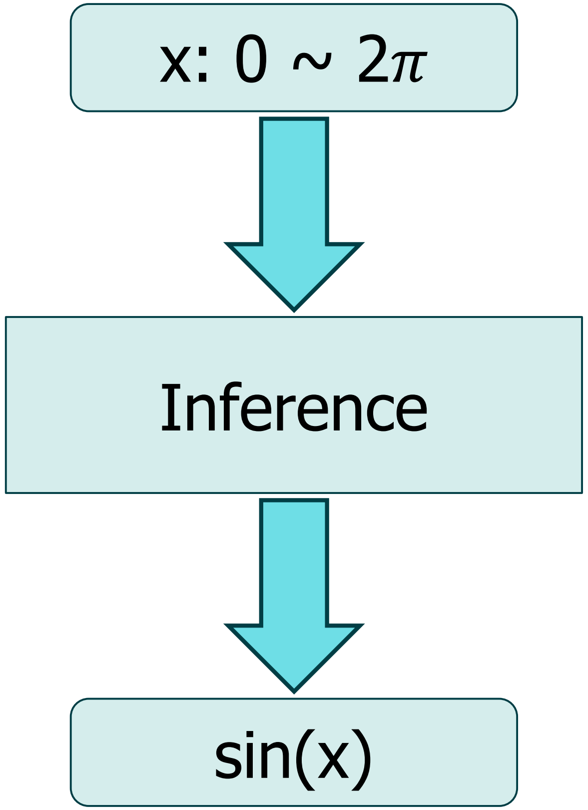
TFLM Arduino Examples: Easy Install

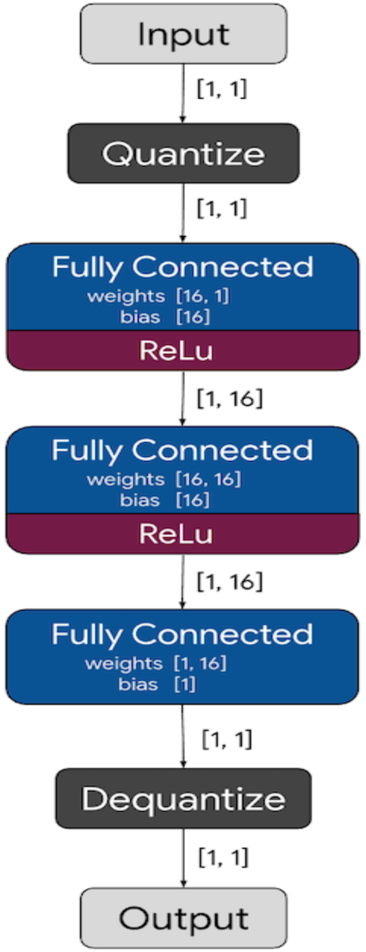
```
File  Edit  View  Search  Terminal  Help

$ mkdir -p ~/Arduino/libraries
$ cd ~/Arduino/libraries
$ git clone https://github.com/tensorflow/tflite-micro-arduino-examples.git
Cloning into 'tflite-micro-arduino-examples'...
remote: Enumerating objects: 2027, done.
remote: Counting objects: 100% (131/131), done.
remote: Compressing objects: 100% (109/109), done.
remote: Total 2027 (delta 55), reused 44 (delta 18), pack-reused 1896
Receiving objects: 100% (2027/2027), 34.89 MiB | 25.00 MiB/s, done.
Resolving deltas: 100% (1199/1199), done.
$
```

download, install and start the Arduino IDE

TFLM Arduino Examples: hello_world





Deeper Dive: Arduino hello_world

The TFLM arena memory contains:

- Modifiable tensors
- Kernel operator execution graph
- Kernel operator and interpreter data structures

The arena memory is statically allocated within the application:

```
constexpr int kTensorArenaSize = 2000;
uint8_t tensor_arena[kTensorArenaSize];
```

Deeper Dive: Arduino hello_world

```
tflite::InitializeTarget();

// Set up logging.
static tflite::MicroErrorReporter micro_error_reporter;
error_reporter = &micro_error_reporter;


// Map the model into a usable data structure. This doesn't involve any
// copying or parsing, it's a very lightweight operation.
model = tflite::GetModel(g_model);


// This pulls in all the operation implementations we need.
static tflite::AllOpsResolver resolver;
```

Deeper Dive: Arduino hello_world

The kernel interpreter needs to be instantiated:

```
// Build an interpreter to run the model with.
static tflite::MicroInterpreter static_interpreter(
    model, resolver, tensor_arena, kTensorArenaSize, error_reporter);
interpreter = &static_interpreter;
```

Then the kernel interpreter initialization and tensor allocation occurs:

```
// Allocate memory from the tensor_arena for the model's tensors.
TfLiteStatus allocate_status = interpreter->AllocateTensors();
```

Deeper Dive: Arduino hello_world

Now we need access to the input tensor so we can fill it with data:

```
// Obtain pointer to the model's input tensor.
input = interpreter->input(0);
```

Since our data is quantized, we need to convert from floating point to int8:

```
// Quantize the input from floating-point to integer
int8_t x_quantized = x / input->params.scale + input->params.zero_point;
// Place the quantized input in the model's input tensor
input->data.int8[0] = x_quantized;
```

Deeper Dive: Arduino hello_world

Time to put the TensorFlow Lite Micro kernel interpreter to work:

```
  // Run inference, and report any error
  TfLiteStatus invoke_status = interpreter->Invoke();
  if (invoke_status != kTfLiteOk) {
    TF_LITE_REPORT_ERROR(error_reporter, "Invoke failed on x: %f\n",
                         static_cast<double>(x));
    return;
  }
```

Deeper Dive: Arduino hello_world

Finally, we get the output tensor so we can see our inference result:

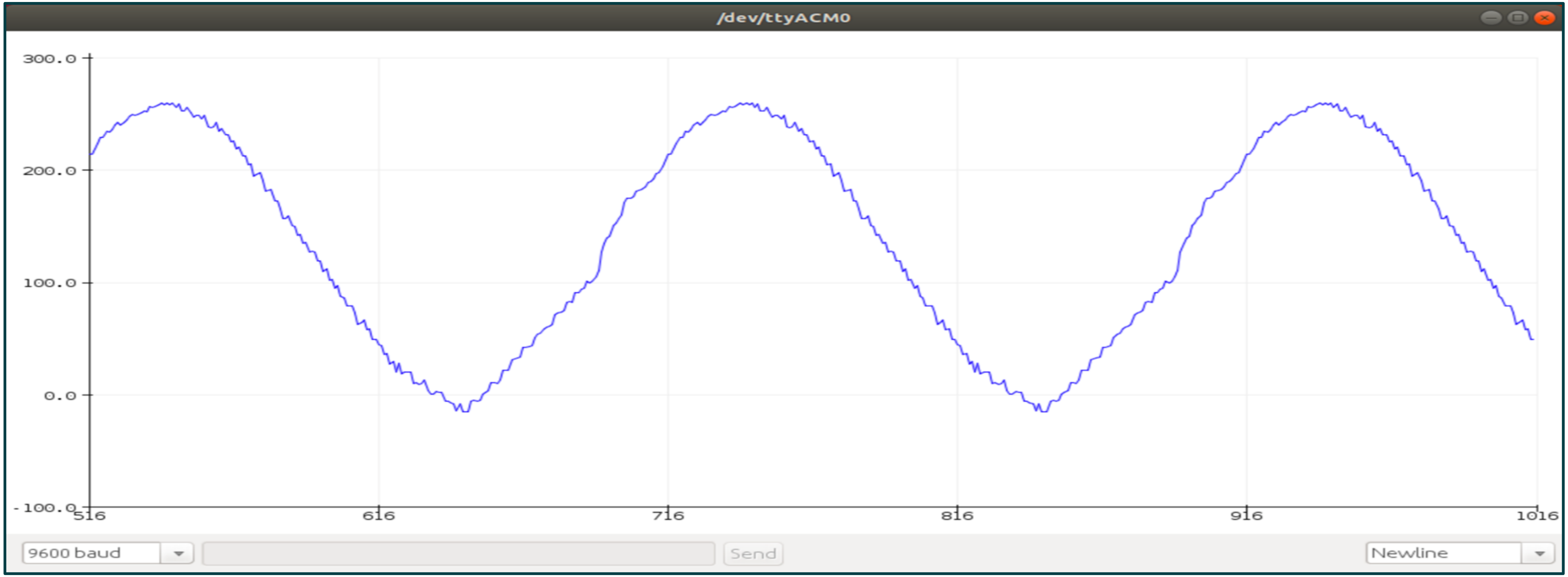
```
  // Obtain pointer to the model's output tensor.
  output = interpreter->output(0);
```

Since our data is quantized, we need to convert it back to floating point:

```
  // Obtain the quantized output from model's output tensor
  int8_t y_quantized = output->data.int8[0];
  // Dequantize the output from integer to floating-point
  float y = (y_quantized - output->params.zero_point) * output->params.scale;
```

Deeper Dive: Arduino hello_world

Arduino IDE serial plotter output:



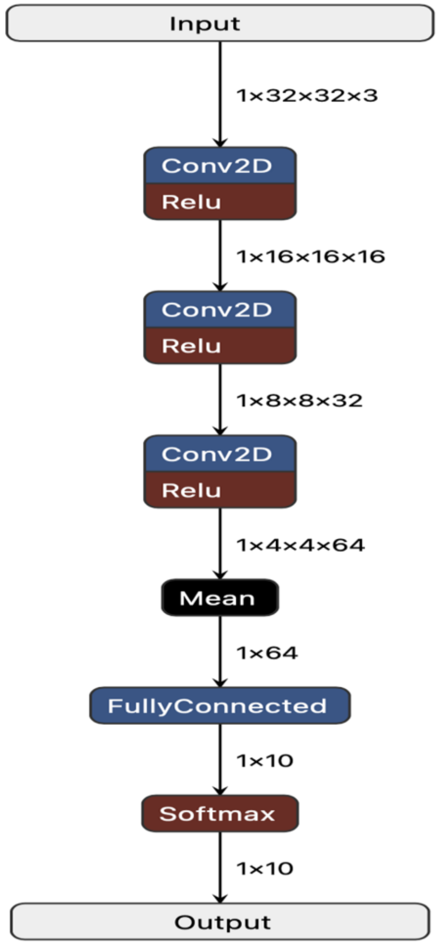
TFLM Arduino Examples: magic_wand

3D IMU Motion Data

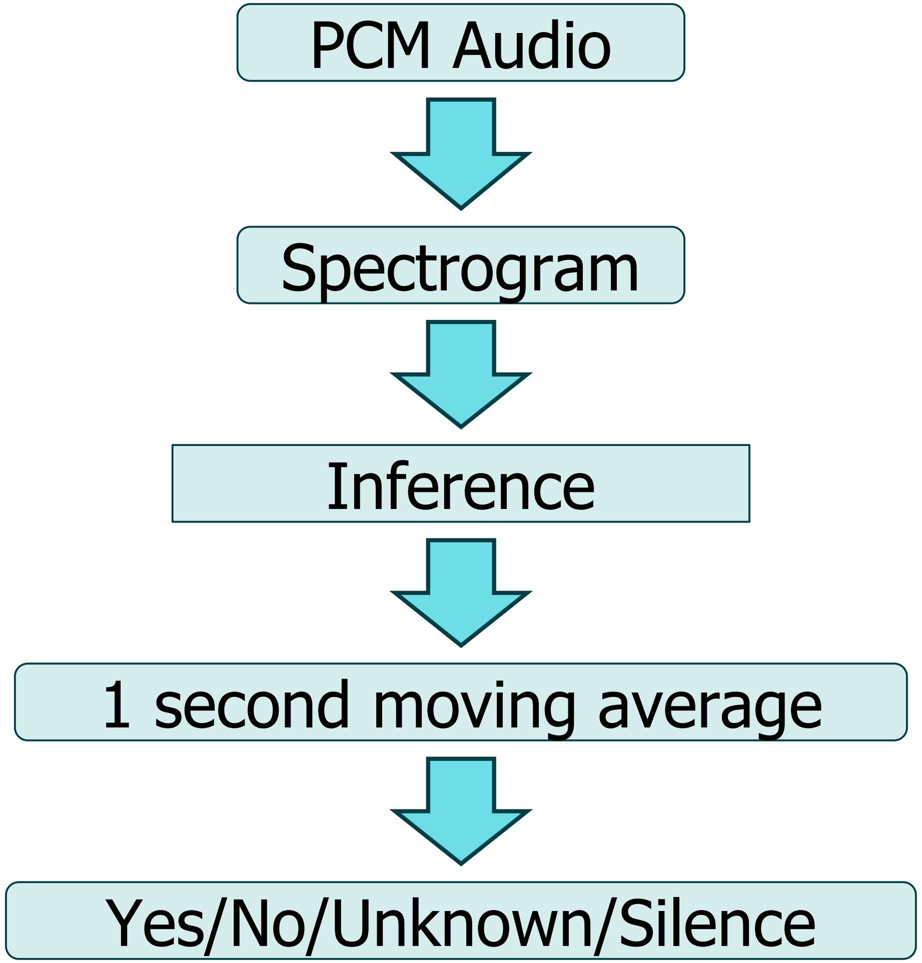
2D Raster Projection (32x32)

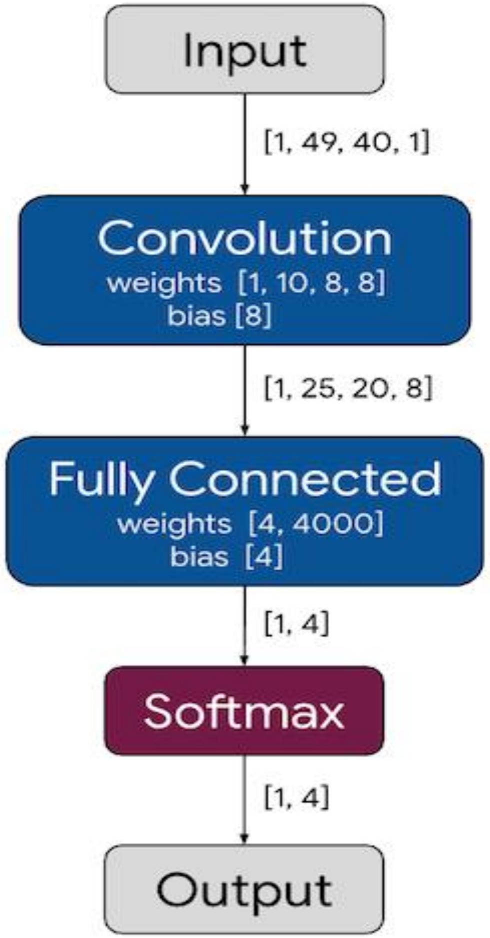
Inference

0-9 Scores

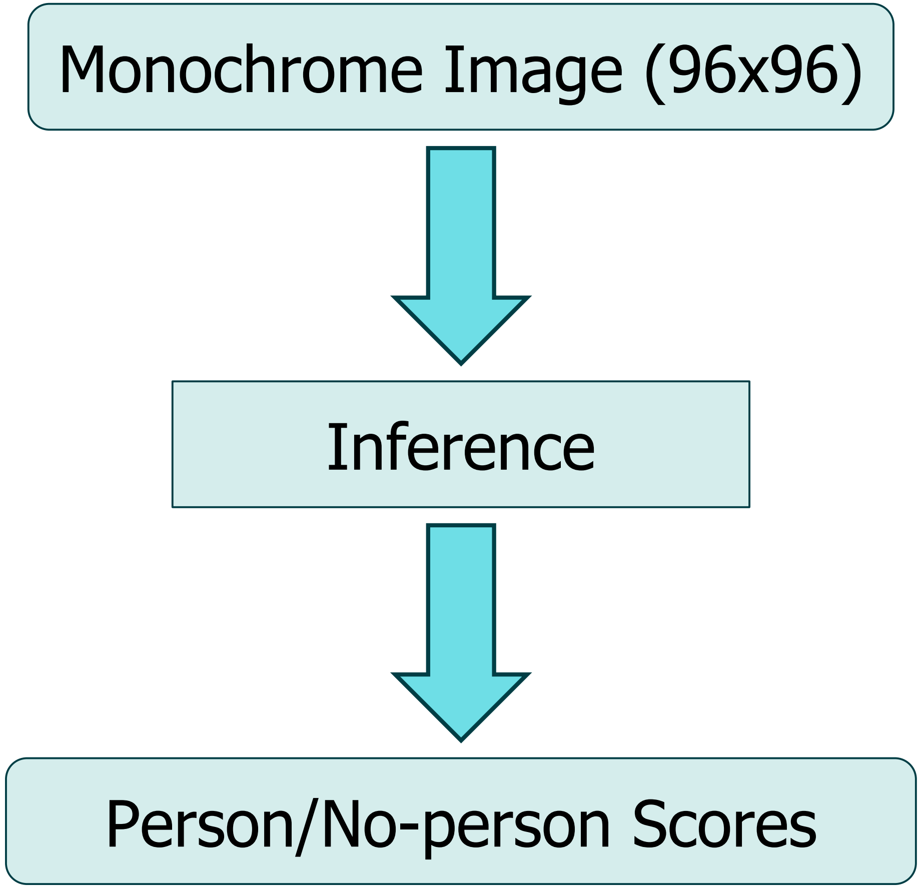


TFLM Arduino Examples: micro_speech





TFLM Arduino Examples: person_detection



- Mobilenet_v1 model
- 31 Kilobytes size
- 470,000 parameters

*Currently supported only in serial test mode

TFLM Arduino Examples: Test Over Serial

BDTI contributed a new module, test-over-serial module:

https://github.com/tensorflow/tflite-micro-arduino-examples/tree/main/src/test_over_serial

This module allows inference data to be supplied to TFLM applications over a serial connection. The module provides application testing, on device, independent of hardware data acquisition.

A Python script sends data specified by a configuration file, and receives inference results from the device. This script is suitable for CI automation.

```
python3 scripts/test_over_serial.py --example person_detection --verbose test
```

BDTI/Google Collaboration: New Kernels with Porting Guide

TFLM Kernel Operators

BDTI ported multiple kernel reference operators to TFLM from TFLite. Float32 and Int8 support are implemented where appropriate.

ADD_N
CAST
CUMSUM
DEPTH_TO_SPACE
DIV
ELU
EXP
EXPAND_DIMS
FILL
FLOOR_DIV
FLOOR_MOD
GATHER
GATHER_ND
L2_POOL_2D
LEAKY_RELU
LOG_SOFTMAX
SPACE_TO_DEPTH

Changing Tensor Shape/Dimensions

To minimize RAM usage, TFLM keeps tensor dimension data in non-volatile memory (flash, ROM, etc). BDTI has contributed a utility function to accommodate kernel operators that need to modify tensor dimensions. The following kernel operators use this function:

- SPACE_TO_DEPTH
- DEPTH_TO_SPACE
- GATHER
- L2_POOL_2D

```
TfLiteStatus CreateWritableTensorDimsWithCopy(TfLiteContext* context,
                                              TfLiteTensor* tensor,
                                              TfLiteEvalTensor* eval_tensor);
```

Kernel Operator Porting Guide

- New porting guide added: https://github.com/tensorflow/tflite-micro/blob/main/tensorflow/lite/micro/docs/porting_reference_ops.md
- Step-by-step explanation with Github Pull Requests from actual kernel operator port
- FAQ added for common questions on memory allocation by kernel operators

BDTI/Google Collaboration:
GitHub Tooling for Continuous Integration

Why GitHub Tooling

- In April 2021 we started refactoring the TFLM code from the TensorFlow repository into a stand-alone TFLM repository.
- Goals for the CI infrastructure included
  - Ability to run tests with various toolchains and a variety of simulated embedded targets
  - Full visibility into the infrastructure for TFLM's community contributors
  - Blueprint of a CI setup that could be replicated and customized for TFLM ports to various hardware and dev boards
  - Reduce the friction in the PR merging process for both contributors and maintainers

CI Components

Github Actions

- Wide range of triggers
- Temporary virtual environment
- Large ecosystem of reusable components

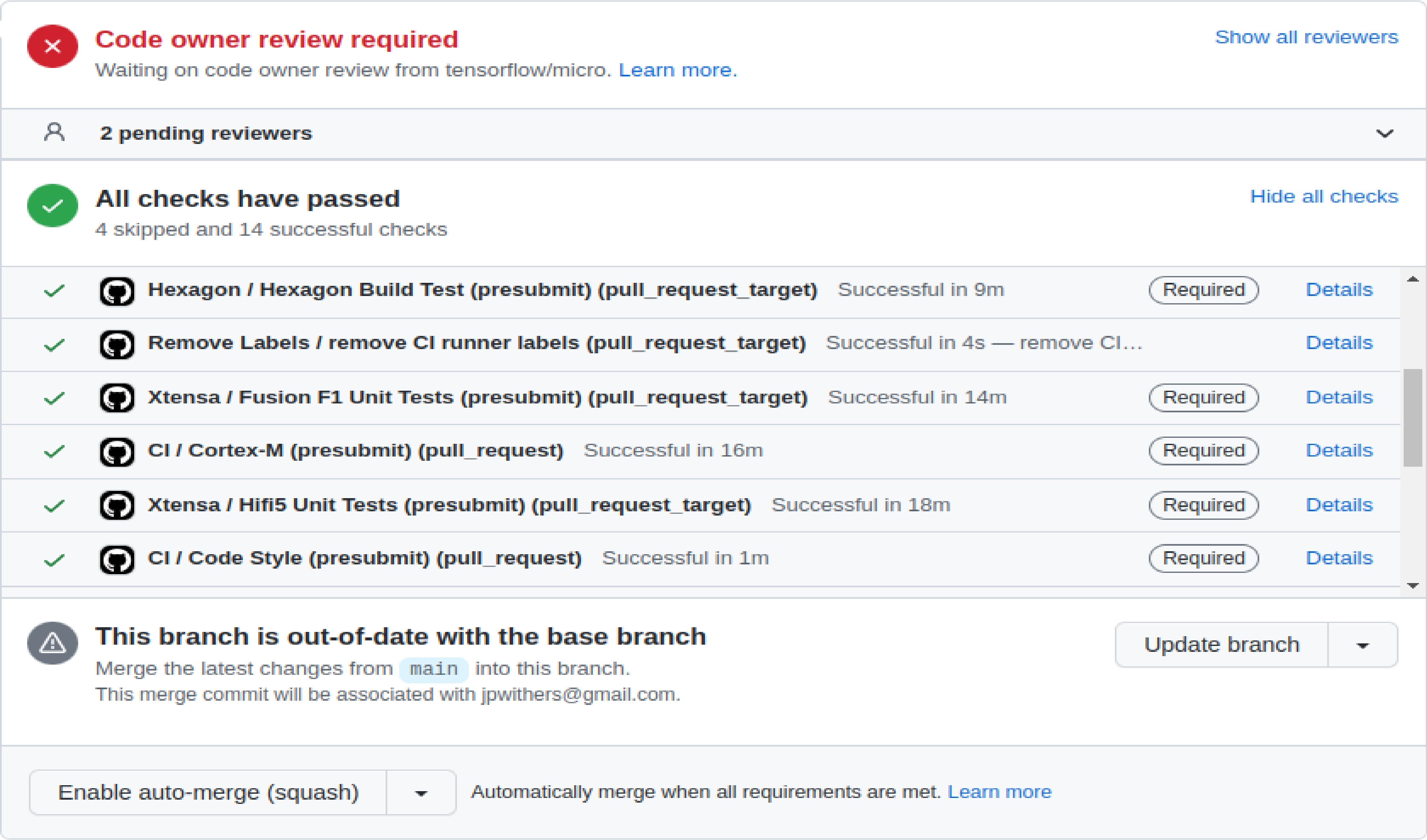
CI Components

GHCR and Docker

- Github Container Repository
- Docker allows easier local testing on developer machines
- Modularizing tests

CI Components

An extensive series of containerized tests are run against the PR



CI Components

Mergify

- Merge queue
- Many PRs end up awaiting reviewer approval
- Continues through merge induced test failures

Developer Story and Subtleties

Developer and Maintainer Story

- Raise a PR to the TFLM repository
- Address reviewer comments
- PR gets merged without additional work from the PR authors
  - E.g., no need to update main when a different PR is merged
- Minimal overhead for the repo maintainers

Subtleties

- Security model for PRs from forks takes a bit of study
- Creative workflows were needed to manage security and community contributions
- Triggers need enhancement

Additional Resources

TensorFlow Lite Micro

https://github.com/tensorflow/tflite-micro

Arduino Examples

https://github.com/tensorflow/tflite-micro-arduino-examples

Contact the TFLM team

https://github.com/tensorflow/tflite-micro#getting-help

2022 Embedded Vision Summit

To see the magic wand demo, stop by the BDTI booth (#413) on the Technology Exhibits floor!

About BDTI

The industry's trusted source for engineering, analysis, and advice for embedded AI, deep learning, and computer vision. Specialties include:

- Algorithm design and implementation
- Processor selection
- Development tool and processor evaluations
- Training and coaching on embedded AI technology

Come see us in Booth 413!