Low-Power Machine Learning

Kirk Benell – SparkFun Electronics
Rob Reynolds – SparkFun Electronics
Scott Hanson – Ambiq Micro
Pete Warden – Google
Presenters

Kirk Benell – SparkFun Electronics
CTO

Rob Reynolds – SparkFun Electronics
Creative Technologist

Scott Hanson - Ambiq Micro
CTO and Founder of Ambiq Micro

Pete Warden - Google
Staff Research Engineering/Lead – TensorFlow Mobile
Goals for Today

Using Machine Learning on a Low-Power Microcontroller

Overall goal is to gain an understanding of TensorFlow Lite on a microcontroller through the following examples:

• Speech Recognition
• Vision Recognition
• Gesture Recognition – “Magic Wand”
• Training a new Machine Learning Model

For the workshop, the Arduino IDE is being used to build, deploy and monitor these examples.
Prerequisites

Software and Hardware required for the workshop

- Computer/Laptop – attendee provided (MacOS, Windows, Linux)
- SparkFun Edge Development Board and accessories – Provided
- The Arduino IDE
  - [https://www.arduino.cc/en/Main/Software](https://www.arduino.cc/en/Main/Software)
- The SparkFun AIOT Example Repository – Contains links to help with setup
- SparkFun Serial Basic Breakout Drivers if needed
  - [https://learn.sparkfun.com/tutorials/sparkfun-serial-basic-ch340c-hookup-guide#drivers-if-you-need-them](https://learn.sparkfun.com/tutorials/sparkfun-serial-basic-ch340c-hookup-guide#drivers-if-you-need-them)
Provided Hardware

- SparkFun Edge Development Board
- SparkFun Serial Basic Breakout (Micro-USB or USB-C)
- USB Cable
- Hi-Max camera for the SparkFun Edge

Note:
- USB-C and Micro-USB cables available
- Let the workshop presenters know if you need another cable
The SparkFun Edge Development Board

Designed for Microcontroller Machine Learning
The results of a collaboration between SparkFun Electronics, Google and Ambiq Micro.

Key Functionality

- Ambiq Micro Apollo3 Blue microcontroller
- Two MEMS microphones
- 3-Axis Accelerometer
- Connector to Interface with a Camera
- Access to SparkFun’s QWIIC device ecosystem
Scott Hanson
CTO and Founder
Ambiq Micro: Intelligence Everywhere

GROUND BREAKING
10X lower power

TRUSTED & PROVEN
>50M units shipped

UNIQUE & DISRUPTIVE
>30 blocking patents

RAPID GROWTH
2X sustained annual revenue growth

Ambiq Micro: Apollo MCU

Copyright © 2019 Arm Dev Summit, All Rights Reserved
Sub-threshold circuits enable >10X energy savings with standard CMOS – enabling >10X more compute without compromising power!
Arm CEO Simon Segars Showcases Ambiq
Extending SPOT to New Markets and New Performance Levels

2017
Apollo2 MCU (+Bluetooth)

2018
Apollo3 MCU (++Memory)

2019
Apollo3 MCU (+AI, audio)

2020
Voyager AI SoC and Solution
Apollo4 MCU (+Display)

Energy Efficiency, Performance
### Selecting the Right SPOT Chip for Your Project

<table>
<thead>
<tr>
<th></th>
<th>Apollo</th>
<th>Apollo2</th>
<th>Apollo2 Blue</th>
<th>Apollo3 Blue</th>
<th>Apollo3 Blue Plus</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>MCU Frequency</strong></td>
<td>24MHz</td>
<td>48 MHz</td>
<td>48MHz</td>
<td>48 MHz, 96 MHz</td>
<td>48 MHz, 96 MHz</td>
</tr>
<tr>
<td><strong>MCU</strong></td>
<td>Arm Cortex M4F</td>
<td>Arm Cortex- M4F</td>
<td>Arm Cortex- M4F Dedicated BLE Core</td>
<td>Arm Cortex- M4F, DMA</td>
<td>Arm Cortex-M0 for BLE</td>
</tr>
<tr>
<td><strong>MCU Power Efficiency</strong></td>
<td>34 uA/MHz</td>
<td>10uA/MHz</td>
<td>10uA/MHz</td>
<td>6uA/MHz</td>
<td>6uA/MHz</td>
</tr>
<tr>
<td><strong>Flash</strong></td>
<td>512KB</td>
<td>1MB</td>
<td>1MB</td>
<td>1MB</td>
<td>2MB</td>
</tr>
<tr>
<td><strong>SRAM</strong></td>
<td>64KB</td>
<td>256KB</td>
<td>256KB</td>
<td>384KB</td>
<td>768KB</td>
</tr>
<tr>
<td><strong>ADC</strong></td>
<td>10 bit, 800 kSps</td>
<td>14 bit, 1.2 MSps</td>
<td>14 bit, 1.2 MSps</td>
<td>14 bit, 1.2 MSps</td>
<td>14 bit, 1.2 MSps</td>
</tr>
<tr>
<td><strong>UART</strong></td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td><strong>I/O</strong></td>
<td>I2C /SPI master</td>
<td>I2C /SPI master (6x)</td>
<td>I2C /SPI masters (4x)</td>
<td>I2C /SPI master (6x)</td>
<td>I2C /SPI master (6x)</td>
</tr>
<tr>
<td><strong>MSPI</strong></td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>QSPI/OSPI Master (1x)</td>
<td>QSPI/OSPI Master (3x)</td>
</tr>
<tr>
<td><strong>I2S</strong></td>
<td>-</td>
<td>-</td>
<td>I2S slave for PDM</td>
<td>I2S slave for PDM</td>
<td>I2S slave for PDM</td>
</tr>
<tr>
<td><strong>PDM</strong></td>
<td>-</td>
<td>Stereo</td>
<td>Stereo</td>
<td>Stereo</td>
<td>Stereo</td>
</tr>
<tr>
<td><strong>Connectivity</strong></td>
<td>-</td>
<td>-</td>
<td>BLE 5</td>
<td>BLE 5</td>
<td>BLE 5</td>
</tr>
<tr>
<td><strong>RF Sensitivity</strong></td>
<td>-</td>
<td>-</td>
<td>-95dBm</td>
<td>-95dBm</td>
<td>-95dBm</td>
</tr>
<tr>
<td><strong>RF TX Max</strong></td>
<td>-</td>
<td>-</td>
<td>+5dBm</td>
<td>+4dBm</td>
<td>+4dBm</td>
</tr>
<tr>
<td><strong>RX current</strong></td>
<td>-</td>
<td>-</td>
<td>3.5mA</td>
<td>3mA</td>
<td>3mA</td>
</tr>
<tr>
<td><strong>TX current</strong></td>
<td>-</td>
<td>-</td>
<td>5.05mA @ 0dBm</td>
<td>3mA @ 0dBm</td>
<td>3mA @ 0dBm</td>
</tr>
<tr>
<td><strong>Packages</strong></td>
<td>- CSP, 27 GPIO</td>
<td>- CSP, 34 GPIO</td>
<td>LGA, 31 GPIO</td>
<td>-CSP, 37 GPIO</td>
<td>BGA, 74 GPIO</td>
</tr>
<tr>
<td></td>
<td>- BGA, 50 GPIO</td>
<td>- BGA, 64 GPIO</td>
<td></td>
<td>-BGA, 50 GPIO</td>
<td></td>
</tr>
</tbody>
</table>

---

arm #ArmDevSummit

Copyright © 2019 Arm Dev Summit, All Rights Reserved
The example code discussed today is meant to kickstart your project quickly – but even simple optimizations beyond this example code can give incredible power/performance.
Machine Learning
A Brief Overview
Machine Learning

A different method to classify information

A traditional method to classify information is based on a deterministic approach – defining a set of rules

Determine if we’ve reached boiling temperature

Temperature Classification

- Temperature $\geq 100\,^\circ C$?
  - Boiling
- Temperature $< 100\,^\circ C$?
  - NOT Boiling
Deep Learning

A method of Machine Learning

Deep Learning is an approach that follows a simple idea of how the human brain works.

Deep Learning uses a network of simulated neurons that are trained to model information relationships.

Networks are made up of layers
Machine Learning

Teaching the Algorithm

For many cases, simple rules are unable to define a result. Machine Learning algorithms solve this by determining the rules through training.

Training a Machine Learning Algorithm

- **Training Data**
  - Data
  - Label

- **Machine Learning Training**
  - Training Model

- **Validate**
  - Result Class = Input Label

- **Feedback**
Machine Learning

Using the Algorithm

The model developed during training is deployed for operational use

The operational machine learning system performs Inference

Using the Machine Learning Algorithm
Pete Warden

TensorFlow Lite - TinyML
Install the TensorFlowLite Library for Arduino

Arduino > Tools > Manage Libraries...

Install Arduino_TensorFlowLite Library

The non-precompiled option
Arduino Setup

Install the SparkFun Board Definitions Package

Open The Arduino Preferences (File->Preferences, (macOS) Arduino > Preferences)

Add the SparkFun Board URL to the “Additional Boards Manager URLs:
https://raw.githubusercontent.com/sparkfun/Ardunio_Boards/master/IDE_Board_Manager/package_sparkfun_index.json
Arduino Setup

Install the SparkFun Board Definitions Package

Arduino > Tools > Board “…” > Manage Boards...

Install the “SparkFun Apollo3 Boards” Package

Select – SparkFun Edge Board

Arduino > Tools > Boards “…” > SparkFun Edge

Set the Bootloader (under Tools) to: “SparkFun Variable Loader”
Connecting the Edge Board

To Connect the Edge Board to your computer

• Connect the Serial to USB Breakout board to the SparkFun Edge
• Connect the Serial-to-USB Breakout to your computer
  • USB-C cables and USB-A are available

The Edge board starts running the Micro Speech example

• Blue LED starts Blinking – system is “operating”
• LED’s flash when the words “Yes” or “No” are detected
Edge Board – Serial Output

Use the Arduino Serial Monitor to Observer the output from the Edge Board

- Start Arduino
- Select the SparkFun Edge board.
  - Tools > Board “…” > SparkFun Edge
- Select the Port the board is connected to
  - Tools > Port > Select the port - Arduino Lists Available Ports
    - **Windows – COM Port** (Use the Windows Device Manager)
    - **macOS – “/dev/cu.usbserial*”** - Note: might have to check permission
    - **Linux – “/dev/ttyUSB*”**
- Once the Port is set, use the Arduino Serial Monitor
  - Tools > Serial Monitor
Examples

Micro Speech
Uses Spectograms for classification

- 2-Dimensional array of frequency data
- FFT of 30ms audio sample – 256 entries
- Every 6 entries averaged -> 43 values
- Values are quantized unsigned 8 bit values

The analysis windows is moved 20ms per sample

- Continues for 49 times
- Creates a 43x49 single band image
Micro Speech – Build and Deploy

Access the Example in Arduino

Arduino > File > Open

Navigate to the Examples micro speech Directory

…/TensorFlow_AIOT2019/micro_speech

Select the “micro_speech.ino” Arduino Sketch
Micro Speech – Code Overview

micro_speech.ino

```cpp
void setup() {
  // The name of this function is important for Arduino compatibility.
  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_tiny_conv_micro_features_model_data);
  if (model->version() != TFLITE_SCHEMA_VERSION) {
    error_reporter->Report(
      "Model provided is schema version %d not equal "
      "to supported version %d."
      , model->version(), TFLITE_SCHEMA_VERSION);
    return;
  }
}
```

setup() - Setup of Sketch

- error_reporter

The model is the result of a training process:

`g_tiny_conv_micro_features_model_data`
Model Data is a static array

- Output of the training process
- Normalized and flattened
- Optimized for micro controller operation

When a new model is created (to change detection words), a similar data set is built and used in the example.
The setup() function

Build Interpeter

Allocate Tensor Memory

Feature Provider for Input

Recognizer for Class Determination

```cpp
static tflite::MicroInterpreter static_interpreter(
    model, micro_mutable_op_resolver, tensor_arena, kTensorArenaSize,
    error_reporter);
    interpreter = &static_interpreter;

// Allocate memory from the tensor_arena for the model's tensors.
TfLiteStatus allocate_status = interpreter->AllocateTensors();
if (allocate_status != kTfLiteOk) {
    error_reporter->Report("AllocateTensors() failed");

static FeatureProvider static_feature_provider(kFeatureElementCount,
    model_input->data.uint8);
    feature_provider = &static_feature_provider;

static RecognizeCommands static_recognizer(error_reporter);
    recognizer = &static_recognizer;
```
Micro Speech – Code Overview

The loop() function

- Fetch the latest Spectogram
- Run the TensorFlow Interpreter
- Determine Feature Recognition
- Output Results

```cpp
TfLiteStatus feature_status = feature_provider->PopulateFeatureData(
    error_reporter, previous_time, current_time, &how_many_new_slices);
if (feature_status != kTfLiteOk) {
    error_reporter->Report("Feature generation failed");
    return;
}

// Run the model on the spectrogram input and make sure it succeeds.
TfLiteStatus invoke_status = interpreter->Invoke();
if (invoke_status != kTfLiteOk) {
    error_reporter->Report("Invoke failed");
    return;
}

bool is_new_command = false;
TfLiteStatus process_status = recognizer->ProcessLatestResults(
    output, current_time, &found_command, &score, &is_new_command);
if (process_status != kTfLiteOk) {
    error_reporter->Report("RecognizeCommands::ProcessLatestResults() failed");
}

// own function for a real application.
RespondToCommand(error_reporter, current_time, found_command, score,
    is_new_command);
```
Micro Speech – Edge Output Driver

`arduino_command_responder.cpp`

### Interface Signature

```cpp
void RespondToCommand(tflite::ErrorReporter* error_reporter,
                      int32_t current_time, const char* found_command,
                      uint8_t score, bool is_new_command) {
  static bool is_initialized = false;

  if (is_new_command) {
    error_reporter->Report("Heard %s (%d) @%dms", found_command, score,
                           current_time);
    // If we hear a command, light up the appropriate LED.

    if (found_command[0] == 'y') {
      last_command_time = current_time;
      digitalWrite(LEDG, LOW); // Green for yes
    }

    if (found_command[0] == 'n') {
      last_command_time = current_time;
      digitalWrite(LED, LOW); // Red for no
    }

    if (found_command[0] == 'u') {
      last_command_time = current_time;
      digitalWrite(LEDB, LOW); // Blue for unknown
    }
  }
}
```

### Notes

If the model changes, the output command values need updating.
Micro Speech – Build and Deploy

Build the Project and Flash the Edge Board

Select the Compile and Upload button. The project is built and then uploaded to the board.

Note – you might see some compiler warnings...
Examples

Person Detection
Input to the Neural Network - Images

This demo uses an image captured by a camera for the classification process

- The model is trained to recognize when a person is present
- When a person is detected, an on-board LED is enabled

For this example, the provided Hi-Max camera is used.

- Make sure the provided camera is connected to the Edge Board (lens facing down)
Person Detector – General Operation

Camera

ImageProvider
Capture Image Data

TF Lite Interpreter
Execute Model

DetectionResponder
Takes action based on person detected

Output

Model
Trained to classify presence or absence of person

loop()
Person Detection—Build and Deploy

Access the Example in Arduino

Arduino > File > Open

Navigate to the Examples micro speech Directory

.../TensorFlow_AIOT2019/person_detection

Select the “person_detection.ino” Arduino Sketch
Person Detection – Loop Details

Setup is very similar to micro_speech

```c
void loop() {
  // Get image from provider.
  if (kTfLiteOk != GetImage(error_reporter, kNumCols, kNumRows, kNumChannels, input->data.uint8)) {
    error_reporter->Report("Image capture failed.");
  }

  // Run the model on this input and make sure it succeeds.
  if (kTfLiteOk != interpreter->Invoke()) {
    error_reporter->Report("Invoke failed.‘");
  }

  TfLiteTensor* output = interpreter->output(0);

  // Process the inference results.
  uint8_t person_score = output->data.uint8[kPersonIndex];
  uint8_t no_person_score = output->data.uint8[kNotAPersonIndex];
  RespondToDetection(error_reporter, person_score, no_person_score);
}
```
Build the Project and Flash the Edge Board

Select the Compile and Upload button. The project is built and then uploaded to the board.

Note – you might see some compiler warnings...
Examples

Magic Wand
Magic Wand – Gesture Recognition

Using accelerometer data to recognize gestures

Uses a 20 KB neural network and TensorFlow Lite to recognize gestures.

Accelerometer data is read from the SparkFun Edge’s on-board accelerometer.

The recognized gesture is output to the serial port and the LED is enabled.
Magic Wand – Gesture Recognition

The Gestures

WING

RING

SLOPE
Magic Wand – Accelerometer Data

Accelerometer data is difficult to classify

A challenge to use heuristics to recognize motion patterns. But deep learning can simplify this effort.

Figure 11-1. Graph showing data for a person who is jogging (from the MotionSense dataset (https://github.com/mmalekzadeh/motion-sense))

Figure 11-2. Graph showing data for a person who is walking downstairs (from the MotionSense dataset (https://github.com/mmalekzadeh/motion-sense))
Magic Wand – Gesture Recognition

General Architecture

Follows a similar pattern to earlier examples.

Uses the 3-axis accelerometer data is read from the SparkFun Edge’s on-board accelerometer (data rate of 25Hz).
Gesture Recognition – General Operation

loop()

Accelerometer Handler
Capture Accel Data

TF Lite Interpreter
Execute Model

Gesture Predictor
Determines if a valid gesture was detected

Output Handler
Takes action based on person detected

Model
Trained to classify three gestures and “unknown”

Accelerometer

Output
Gesture Recognition – Build and Deploy

Access the Example in Arduino
Arduino > File > Open

Navigate to the Examples micro speech Directory
	.../TensorFlow_AIoT2019/gesture_recognition

Select the “gesture_recognition.ino” Arduino Sketch
Gesture Recognition—Loop Details

Setup is very similar to previous examples

```c
void loop() {
    // Attempt to read new data from the accelerometer
    bool got_data = ReadAccelerometer(error_reporter, model_input->data.f,
                                       input_length, should_clear_buffer);

    // Run inference, and report any error
    TFLiteStatus invoke_status = interpreter->Invoke();
    if (invoke_status != kTFLiteOk) {
        error_reporter->Report("Invoke failed on index: %d\n", begin_index);
        return;
    }

    // Analyze the results to obtain a prediction
    int gesture_index = PredictGesture(interpreter->output(0)->data.f);
    // Clear the buffer next time up-reading data

    // Produce an output
    HandleOutput(error_reporter, gesture_index);
```
Build the Project and Flash the Edge Board

Select the Compile and Upload button. The project is built and then uploaded to the board.

Once Flashed – Grab your board and try the Gestures...
Magic Wand – Gesture Recognition

The Gestures

- **WING**
- **RING**
- **SLOPE**

HOLD THE EDGE FACING YOU
Training Example

Training a new Speech Model
Addition Example – Training a Model

Training a new Speech Model using Google Colaborator

Google Colaborator is a Jupyter Notebook based method to run Python (iPython Notebooks).

Abstracts the hardware requirements needed for ML training, focusing on executing a notebook.

The notebook for this is contained in the TensorFlow repository:

tensorflow/lite/experimental/micro/examples/micro_speech/train_speech_model.ipynb
The TensorFlow provided notebook is very well documented.

Just change the value of the “WANTED_WORDS”

Once the training process is complete, the resulting model can be downloaded and used in the micro_speech demo.

Execution using GPU runtime takes 1.5 – 2 hours.
Replace the Feature Data with the results of the training session.

The training data is contained in the file tiny_conv.cc

Update the static array contents.
Micro Speech – Update Output Routine

arduino_command_responder.cpp

```cpp
if (found_command[0] == 'y') {
    last_command_time = current_time;
    digitalWrite(LEDG, LOW); // Green for yes
}

if (found_command[0] == 'n') {
    last_command_time = current_time;
    digitalWrite(LEDR, LOW); // Red for no
}

if (found_command[0] == 'u') {
    last_command_time = current_time;
    digitalWrite(LEDB, LOW); // Blue for unknown
}
```

Update with the first letter of the keywords being detected.
Update the class name array. Order matters in this list.

```cpp
const char* kCategoryLabels[kCategoryCount] = {
    "silence",
    "unknown",
    "yes",
    "no",
};
```
Recap – The Goals for Today

Using Machine Learning on a Low-Power Microcontroller

Overall goal is to gain an understanding of TensorFlow Lite on a microcontroller through the following examples:

• Speech Recognition
• Vision Recognition
• Gesture Recognition – “Magic Wand”
• Training a new Machine Learning Model
THANK YOU!